

**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE
MINAS GERAIS**

**CAMPUS DIVINÓPOLIS
GRADUAÇÃO EM ENGENHARIA MECATRÔNICA**

Artur Alves Pinto

**PROCESSAMENTO DE SINAIS SONOROS PARA
IDENTIFICAÇÃO DE NOTAS MUSICAIIS UTILIZANDO O
RASPBERRY PI**

Divinópolis
2018

Artur Alves Pinto

**PROCESSAMENTO DE SINAIS SONOROS PARA
IDENTIFICAÇÃO DE NOTAS MUSICAIIS UTILIZANDO O
RASPBERRY PI**

Monografia de Trabalho de Conclusão de Curso apresentado ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Eletrônica, Programação e Sinais e Sistemas.

Orientador: Prof. Dr. Emerson de Sousa Costa
Co-orientador: Prof. Dr. Luís Filipe Pereira Silva.

Co-orientador: Mestre Marcos Alberto Saldanha.

Divinópolis
2018

(Catalogação - Biblioteca Universitária – Campus Divinópolis – CEFET-MG)

P659p Pinto, Artur Alves.

Processamento de sinais sonoros para identificação de notas musicais
utilizando o *Raspberry PI*. / Artur Alves Pinto. - Divinópolis, 2018.

98f. : il.

Orientador: Prof. Dr. Emerson de Sousa Costa.

Coorientador: Prof. Dr. Luis Filipe Pereira Silva.

Mestre Marcos Alberto Saldanha.

Trabalho de Conclusão de Curso (graduação) – Colegiado de
Graduação em Engenharia Mecatrônica do Centro Federal de Educação
Tecnológica de Minas.

1. Eletrônica.
 2. Programação.
 3. Sinais e Sistemas.
 4. *Raspberry PI*
 5. Música.
 6. Novas Tecnologias.
 7. *Python*.
- I. Costa, Emerson de Sousa.
II. Silva, Luis Filipe Pereira. III. Saldanha, Marcos Alberto. IV. Centro
Federal de Educação Tecnológica de Minas Gerais. V. Título.

CDU: 62(043)

Artur Alves Pinto

PROCESSAMENTO DE SINAIS SONOROS PARA IDENTIFICAÇÃO DE NOTAS MUSICais UTILIZANDO O RASPBERRY PI

Monografia de Trabalho de Conclusão de Curso apresentado ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.

Áreas de integração: Eletrônica (Amplificador Operacional, Filtro Ativo, Conversor A/D), Programação (*Python*) e Sinais e Sistemas (*Fast Fourier Transform*, Digitalização de Sinal).

Comissão Avaliadora

Prof. Dr. Emerson de Sousa Costa (Orientador)

Centro Federal de Educação Tecnológica de Minas Gerais - CEFET MG

Prof. Dr. Renato de Sousa Dâmaso

Centro Federal de Educação Tecnológica de Minas Gerais - CEFET MG

Prof. Dr. Cláudio Henrique Gomes dos Santos

Centro Federal de Educação Tecnológica de Minas Gerais - CEFET MG

Divinópolis
2018

*Dedico este trabalho à Deus, porque tudo vem dEle e à minha mãe,
minha avó e minha noiva, pelo apoio e confiança.*

AGRADECIMENTOS

Agradeço primeiramente a Deus, o qual me capacitou e me permitiu passar por esta experiência e alcançar o sucesso nos estudos. Sempre pude contar com a ajuda dEle.

Agradeço também à minha mãe, Sidney Alves de Assis, pelo apoio em todos os momentos de dificuldades e por nunca ter deixado me faltar nada.

Sou grato a minha avó, Benedita de Sá, que sempre me incentivou a buscar meus sonhos.

Agradeço a minha noiva, Rafaela Petuel Oliveira de Paula, que nunca deixou de acreditar na minha capacidade, acompanhou de perto tudo que passei e sempre esteve ao meu lado quando precisei.

Sou grato aos meus amigos Diego Santos e Áquila Garcia pelo incentivo e companheirismo.

Agradeço a meu amigo Lucas José pelo apoio com a moradia e por não me deixar desistir e aos meus amigos Marcos Alberto, Leonardo Maciel e Fernando Martins pela ajuda ao longo do curso e também no desenvolvimento deste trabalho.

Por último, não menos importante, agradeço ao meu orientador Prof. Dr. Emerson de Sousa pela disposição, atenção e auxílio neste trabalho.

“Try not. Do... or do not. There is no try.”
(Master Yoda – Star Wars)

RESUMO

Este Trabalho de Conclusão de Curso (TCC) visa realizar a leitura de um sinal sonoro analógico, emitido por um instrumento musical e, através de um equipamento de baixo custo, identificar a nota musical correspondente a este sinal para uma ampla faixa de frequência. Além disso, busca-se avaliar as possíveis aplicações de novas tecnologias para o avanço benéfico da música. Para isso, utiliza-se o minicomputador Raspberry Pi 3 (RPI3) como plataforma para processar o sinal e gerar uma interface de trabalho. O RPI3 é um minicomputador novo no mercado, foi lançado em fevereiro de 2016 e por isso as aplicações para esse dispositivo são bem restritas e estão em pleno desenvolvimento. Sua utilização no meio musical ainda é pouco estudada, mas sabe-se que possui características que o tornam útil para realização de algumas tarefas, como processar sinais, ser programável para implementação de diversas funções, entre outras opções. Como o avanço da tecnologia tem se tornado cada vez mais presente na música, o RPI3 pode vir a ser uma boa opção para implementações e possíveis inovações na música, uma vez que possui um baixo custo e pode ser facilmente programado para diferentes tarefas através da linguagem Python (linguagem de alto nível). Com base nisto, este projeto busca implementar um circuito eletrônico, que será acoplado ao RPI3 para aquisição de dados analógicos e a conversão dos mesmos para digital. Assim estes dados digitais podem ser acessados a partir das portas de saída/entrada digitais (GPIO) do RPI3 e trabalhados por meio de uma linguagem de programação de alto nível, disponibilizada pelo sistema operacional do RPI3, conhecido por Raspbian. É possível ainda gerar a interface gráfica para o trabalho com programas disponibilizados pelo Raspbian, que podem ser utilizados para mostrar em uma tela as notas musicais correspondentes ao sinal sonoro captado. Isto torna este trabalho autêntico, trazendo um estudo para aplicação de uma nova tecnologia no meio musical e consequentemente abre uma nova possibilidade de pesquisa e projetos dentro dessa área.

Palavras-chave: Sinais, Rasberry Pi 3, Novas Tecnologias, Música.

ABSTRACT

This Course Completion Work (TCC) aims to perform the reading of an analog sound signal, emitted by a musical instrument and, through a low cost equipment, identify the musical note corresponding to this signal for a wide frequency range. In addition, it seeks to evaluate the possible applications of new technologies for the beneficial advancement of music. For this, the minicomputer Raspberry is used Pi 3 (RPI3) as platform to process the signal and generate a working interface. The RPI3 is a new minicomputer in the market, was released in February 2016 and therefore the applications for this device are very restricted and are in full development. Its use in the musical environment is still little studied, but it is known that it has characteristics that make it useful for performing some tasks, such as processing signals, being programmable to implement various functions, among other options. As the advancement of technology has become increasingly present in music, RPI3 can be a good choice for implementations and possible innovations in music, since it has a low cost and can be easily programmed for different tasks through language Python (high level language). Based on this, this project seeks to implement an electronic circuit, which will be coupled to RPI3 to acquire analog data and convert them to digital. Thus, these digital data can be accessed from RPI3's digital I / O (GPIO) ports and worked through a high-level programming language provided by the RPI3 operating system known as Raspbian. It is also possible to generate the graphical interface for the work with programs made available by Raspbian, which can be used to display on a screen the musical notes corresponding to the sound signal picked up. This makes this work authentic, bringing a study to apply a new technology in the musical environment and consequently opens a new possibility of research and projects within this area.

Keywords: Signs, Rasberry Pi 3, New Technologies, Music

SUMÁRIO

1. Introdução.....	1
1.1. Definição do Problema	3
1.2. Motivação	3
1.3. Objetivo do Trabalho	4
1.4. Estado da Arte.....	5
1.5. Organização do Documento	6
2. Fundamentação Teórica	7
2.1. Revisão de Literatura.....	7
2.2.1. Sistema Operacional	11
2.2.2. Portas GPIO (General Purpose Inputs and Outputs)	12
2.2.3. Programação em Python	14
2.3. Análise de Sinais	14
2.3.1. Sinais Contínuos e Discretos no Tempo.....	15
2.3.2. Sinais Analógicos e Sinais Digitais	15
2.3.3. Sinais Periódicos e Não Periódicos	16
2.3.4. Ruído.....	17
2.3.5. Sistemas digitais e Sistemas Analógicos	17
2.4. Processamento Digital de Sinais	18
2.4.1. Teorema da Amostragem e Nyquist	19
2.4.2. <i>Aliasing</i>	20
2.4.3. Reconstrução de Sinal (Retentor de Ordem Zero).....	21
2.5. Eletrônica de Sistemas	23
2.5.1. Transdutores	23
2.5.2. Conversor Analógico-Digital (A/D)	25
2.5.3. Filtros	25
2.6. Fundamentação Musical.....	28
2.6.1. Notas Musicais	28
2.6.2. Percepção de um Som	29
2.6.3. Parciais Harmônicas.....	32
2.6.4. Computação e Música	33
2.7. <i>Fast Fourier Transform (FFT's)</i>	34

3. Metodologia	39
3.1. Inicialização do Raspberry PI 3.....	39
3.1.1. Instalação do Sistema Operacional e Configurações Iniciais	40
3.1.2. Configurações das Portas GPIO.....	42
3.1.3. Configuração da Interface SPI.....	43
3.2. Determinação da Faixa de Frequência	47
3.3. Circuito de Aquisição de Dados	48
3.3.1. Amplificador Operacional LM386	51
3.3.2. Filtro Ativo de 3 ^a Ordem.....	53
3.4. Código Desenvolvido para Aplicação de FFT e Identificação da Nota Musical.	
55	
4. Resultados.....	58
4.1. Mapeamento de Frequências	58
4.2. Captação de Áudio	59
4.2.1. Circuito de ganho e off-set.....	60
4.2.2. Filtro Ativo de 3 ^a Ordem.....	63
4.3. Reconstrução do sinal	66
5. Conclusões e Sugestões para Trabalhos Futuros.....	72
5.1. Conclusões.....	72
5.2. Sugestões para Trabalhos Futuros.....	73
REFERÊNCIAS.....	75
APÊNDICE A.....	79
APÊNDICE B.....	80
APÊNDICE C.....	83

LISTA DE FIGURAS

Figura 2. 1: RASPBERRY PI 3.....	10
Figura 2. 2: FUNÇÕES DA GPIO DO RPI3.	13
Figura 2.3: SINAL CONTÍNUO X SINAL DISCRETO.....	15
Figura 2. 4: (a) SINAL ANALÓGICO; (b) SINAL DIGITAL	16
Figura 2. 5: (a) SINAL CONTÍNUO NO TEMPO $x(t)$; (b, c, d) AMOSTRAGENS COM n AMOSTRAS DISCRETAS NO TEMPO DO SINAL $x(t)$	19
Figura 2. 6: EFEITO DO ALIASING EM UM SINAL.....	20
Figura 2. 7: FORMA DE ONDA GERADA PELO ZOH.	22
Figura 2. 8: MICROFONE DE ELETRETO.	24
Figura 2. 9: INTERIOR DE UM MICROFONE DE ELETRETO; (a) VISTA LATERAL; (b) VISTA FRONTAL.....	24
Figura 2. 10: (a)PASSA-BAIXA; (b)PASSA-ALTA; (c)PASSA-FAIXA.	26
Figura 2. 11: CIRCUITO RC - FILTRO PASSA-BAIXA.....	27
Figura 2. 12: CIRCUITO RC - FILTRO PASSA-ALTA.	27
Figura 2. 13: CIRCUITO RLC - PASSA-FAIXA.....	28
Figura 2. 14: FORMA DE ONDA SENOIDAL DE 400Hz.....	30
Figura 2. 15: FORMAS DE ONDAS DE DIFERENTES INSTRUMENTOS PARA UMA MESMA NOTA MUSICAL.....	31
Figura 2. 16: ETAPAS DE UMA ONDA SONORA EMITIDA POR UM INSTRUMENTO AO LONGO DO TEMPO.	31
Figura 2. 17: ESPECTRO DE FREQUÊNCIAS FUNDAMENTAL E HAMÔNICAS DE UMA NOTA A4.	32
Figura 2. 18: FUNÇÃO DELTA DIRAC (IMPULSO) DESLOCADO NO TEMPO.	35
Figura 2. 19: REPRESENTAÇÃO DE UM SINAL (TEMPO) E SUA RESPECTIVA TRANSFORMADA DE FOURIER (FREQUÊNCIA).	36
Figura 2. 20: PROCESSO DE JANELAMENTO DE UM ESPECTRO DE SINAL.	38

Figura 3.1: BANCADA DE TRABALHO PARA INICIALIZAÇÃO DO RPI3.....	40
Figura 3.2: CONFIGURAÇÃO DOS PINOS GPIO.....	43
Figura 3.3: LIGAÇÃO MCP3008 AO RASPBERRY PI3.....	44
Figura 3.4: TELA DE CONFIGURAÇÃO DE SOFTWARE DO RASPBERR PI	44
Figura 3.5: TELA DE CONFIGURAÇÃO PARA ATIVAÇÃO DA INTERFACE SPI.....	45
Figura 3.6: SELECIONE YES PARA REINICIALIZAR O RPI.....	45
Figura 3.7: LINHAS DE CÓDIGO PARA INSTALAÇÃO DA BIBLIOTECA ADAFRUIT PYTHON MCP3008.....	46
Figura 3.8: TELA APÓS A INSTALAÇÃO BEM-SUCEDIDA DA BIBLIOTECA ADAFRUIT PYTHON MCP3008.	46
Figura 3.9: EXEMPLO DE LEITURA DO CONVERSOR MCP3008.....	47
Figura 3.10: RELAÇÃO DAS NOTAS MUSICAIS EM UMA OITAVA.....	48
Figura 3.11: (a) MONTAGEM ELÉTRICA DA POLARIZAÇÃO DO MICROFONE. (b) ESQUEMÁTICO DE POLARIZAÇÃO DO MICROFONE.....	49
Figura 3.12: CIRCUITO GERADOR DE GANHO E OFF-SET NA ONDA.....	50
Figura 3.13: LM386 E SUAS ETAPAS INTERNAS PARA TRATAMENTO DO SINAL.....	51
Figura 3.14: CIRCUITO DO MODO NÃO OPERACIONAL PARA CÁLCULO DE GANHO DE TENSÃO.....	52
Figura 3.15: FAIXA DE PASSAGEM E FAIXA DE BLOQUEIO, FORMANDO A BANDA PASSANTE.	54
Figura 3.16: COMPARAÇÃO DE RESPOTA DOS MODELOS BUTTHERWORTH E CHEBYSHEV.	55
Figura 3.17: FILTRO ATIVO PASSA BAIXA DE BUTTERWORTH.....	55
Figura 3.18: FLUXOGRAMA REPRESENTANDO A LÓGICA DO CÓDIGO DESENVOLVIDO.	56
Figura 4.1: FORMA DE ONDA DO DESACOPLAMENTO E POLARIZAÇÃO DO MICRFONE.	59
Figura 4.2: AMBIENTE DE SIMULAÇÃO UTILIZADO PARA TESTES NO PROTEUS 8.....	60
Figura 4.3: RESPOTA CC OBTIDA NA SIMULAÇÃO.	61
Figura 4.4: RESPOSTA CA OBTIDA NA SIMULAÇÃO.....	61

Figura 4.5: RESPOSTA DO CIRCUITO REAL QUANDO SUBMETIDO A UM SINAL DE 1KHZ.	62
Figura 4.6: DIAGRAMA DE BODE DO FILTRO PROJETADO.	63
Figura 4.7: ATRASO DO SINAL QUANDO PASSA PELA FILTRAGEM.	64
Figura 4.8: RESPOSTA DO CIRCUITO GANHO, OFF-SER E FILTRO PARA UMA ENTRADA DE 2KHZ	64
Figura 4.9: RESPOSTA DO CIRCUITO GANHO, OFF-SER E FILTRO PARA UMA ENTRADA DE 4KHZ	65
Figura 4.10: RESPOSTA DO CIRCUITO GANHO, OFF-SER E FILTRO PARA UMA ENTRADA DE 8KHZ	66
Figura 4.11: PARÂMETROS PRÉ-ESTABELECIDOS PARA EXECUÇÃO DO CÓDIGO E EVENTUALMENTE DO TESTE	67
Figura 4.12: RECONSTRUÇÃO DA FREQUÊNCIA DE 500HZ.	67
Figura 4.13: RECONSTRUÇÃO DA FREQUÊNCIA DE 4000HZ.	68
Figura 4.14: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA E2	69
Figura 4.15: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA A2	69
Figura 4.16: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA D3	70
Figura 4.17: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA G3	70
Figura 4.18: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA B3	70
Figura 4.19: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA E4	71

LISTA DE TABELAS

Tabela 2.1: INFORMAÇÕES TÉCNICAS DO RPI3.....	10
Tabela 4.1: NOTAS MUSICAIS E SUAS FREQUÊNCIAS EM UMA FAIXA DE 7 OITAVAS. ...	58

LISTA DE ACRÓNIMOS

RPI(3)	Raspberry Pi (Modelo 3)
GPIO	<i>General Purpose Input Output</i> (Entrada e Saída de Uso Geral)
A/D	Analógico para Digital
G_V	Ganho de Tensão
A	Ganho em Decibéis (dB)
f_0	Frequência Fundamental
f_s	Frequência de Amostragem
T_0	Período Fundamental
P_s	Tempo de Amostra
T	Período de Amostragem
ω_c	Frequência de Corte
T	Tom
ST	Semi-Tom
CI	Circuito Integrado
FFT	Transformada Rápida de Fourier

1. Introdução

Atualmente, a música está atrelada ao cotidiano de todos, diariamente. Mesmo uma pessoa que não sabe tocar ou que não tem conhecimento da teoria musical sabe o nome das notas que compreendem a escala musical, ou gosta de aproveitar os sons provenientes pelo solo de um instrumento ou ainda os efeitos existentes na música eletrônica. No entanto, a música não se limita apenas a notas, instrumentos musicais e diversidade rítmica. Ela vai muito além da emissão de um simples som. Quando uma nota (só uma corda do violão, por exemplo) ou acorde (no mínimo 3 notas ao mesmo tempo) é tocado emite-se uma determinada onda sonora (vibração) com frequência bem definida, a qual gera uma alteração na pressão do ar em um ambiente, as quais são captadas pelo ouvido e transmitidas ao cérebro por meio de diferentes impulsos nervosos (ALEIXO, 2003). Essa geração de vibrações remete a vários estudos para entender e descrever esse efeito físico.

A música é uma arte capaz de levar seu ouvinte à alguma expressão, compreensão ou manifestação (SEKEFF, 2007, p.17). Isso acontece por meio de diversos instrumentos capazes de gerar um som, os quais seguem um mesmo tom, ritmo e harmonia. Ela carrega consigo uma série de informações, desde a física da onda de uma nota, a estrutura de um instrumento, as rimas e fonética de uma letra, até a capacidade de gerar várias expressões emocionais em uma pessoa. Toda essa diversidade faz com que a música englobe várias áreas de estudo, como por exemplo, Matemática, Física, Psicologia, História, Letras, entre outras.

Com o avanço acelerado da tecnologia na última década e devido à relação da música com a ciência, pode-se perceber que a eletrônica e a computação vêm sendo cada vez mais presente no âmbito musical. Grande parte do avanço da música se deve as possibilidades disponibilizadas por estas duas grandes áreas. A eletrônica está presente por trabalhar, principalmente, formas de amplificar, captar e renderizar o som, além de permitir a criação de efeitos e modulação de diferentes timbres, e de novos instrumentos como mixers,

sintetizadores, controladores, entre outros. A computação, por sua vez, contribui com digitalização de músicas, tornando-as de fácil acesso, permitindo o seu armazenamento e também contribui na programação de instrumentos musicais virtuais além de softwares próprios para edição, gravação entre outras tarefas (SERRA, 2002).

Atualmente, é notório o aumento do número de computadores usados por parte dos músicos e editores de áudio. Isso infere ao fato de que a tecnologia e seus avanços tem impacto direto no ambiente musical. Os computadores permitem realizar diversas tarefas que colaboram tanto com a preservação e qualidade da música como a possibilidade de implementações de novos projetos que possam contribuir com o avanço e desenvolvimento da ciência por trás da música.

Assim, o interesse deste trabalho é o desenvolvimento e a avaliação de uma aplicação no âmbito musical para o Raspberry Pi 3, um minicomputador que tem sido visado por sua potência e baixo custo, mas que ainda possui poucos estudos acerca de suas implementações. A ideia veio da possibilidade de usar esse minicomputador para registros de partituras em *real-time*, pois, apesar de toda inovação tecnológica, registrar partituras ainda é uma tarefa feita pelo músico e, na maioria das vezes, leva tempo e torna-se cansativa. Identificar notas ou acordes musicais é o primeiro passo para conseguir realizar esta tarefa. Portanto, este trabalho está voltado para realização da captação de sons musicais monofônicos e identificação dos mesmos em relação a frequência gerada e à nota musical correspondente à esta frequência, além de avaliar a viabilidade de realizar processamento digital utilizando o RPI3.

1.1. Definição do Problema

O problema se consiste na captação e reconstrução digital de um sinal sonoro analógico, de forma que este sinal possa ter uma qualidade considerável e, deste modo, seja possível ter acesso às suas informações e características, com o objetivo de realizar a identificação da nota musical correspondente ao som emitido pelo instrumento. Como o RPI3 não possui entradas analógicas, um circuito com um CI para conversão A/D foi elaborado para a aquisição de dados e consequentemente servirá como ponte de contato entre o microfone e minicomputador. Além disso será necessário criar uma interface gráfica para visualizar as notas correspondentes. Este trabalho envolve as áreas de Sinais e Sistemas, Eletrônica e Programação.

Os conhecimentos de eletrônica são usados para construção do circuito de aquisição e digitalização do sinal sonoro analógico, além de utilizar do conceito de filtros para possíveis na qualidade do sinal captado. A computação é usada na programação realizada em Python e criação da interface. Por último e o mais importante, os conhecimentos de Sinais e Sistema são utilizados para todo entendimento do processo, desde o funcionamento simples de um conversor A/D até os conceitos necessários para efetuar a reconstrução de um sinal analógico, além disso é utilizado aplicações de *Fast Fourier Transform* (FFT) como meio de identificação das frequências musicais.

1.2. Motivação

O que resultou na escolha do tema proposto foi o interesse do autor pelas interferências tecnológicas no âmbito musical, uma vez que estas trouxeram muitos benefícios para o desenvolvimento da música com o passar dos anos. Com o decorrer dos estudos na graduação foi sendo cada vez mais notória a ligação entre Matemática, Eletrônica, Controle e Programação com o avanço musical da última década.

Assim, surgiu o interesse em se aprofundar nestas áreas dando um foco maior nas aplicações que cada uma delas poderia possuir no âmbito musical, fosse para tratamento de som, inovação de algum instrumento ou

processamento de sinais. Além disto, o tema abrangido por este trabalho é pouco abordado durante a graduação, então desperta-se o interesse em realizar o desenvolvimento de um projeto que possa acrescentar academicamente e incentivar outras pesquisas e trabalhos envolvendo Engenharia e música. Portanto, espera-se que este projeto resulte em ideias e aplicações futuras não só para o RPI3, como também para outros dispositivos que possam acrescentar na ciência e física de um som, música ou em novos instrumentos musicais.

1.3. Objetivo do Trabalho

O objetivo geral deste Trabalho de Conclusão de Curso é realizar a captação e digitalização de um sinal sonoro analógico emitido por um instrumento musical e, a partir disto, ser capaz de obter informações sobre as características qualitativas relativas à forma de onda do sinal de entrada. Os objetivos específicos são enumerados a seguir:

1. Projetar e desenvolver um circuito eletrônico para realizar a aquisição e digitalização do sinal analógico.
2. Instalar o Raspbian, configurar e montar o minicomputador Raspberry Pi.
3. Estudar as características físicas da forma de onda de um sinal sonoro e quais são suas relações com a música.
4. Avaliar a necessidade de projetar um amplificador e ou filtro para melhoria de qualidade do sinal de entrada.
5. Estudar como manipular os dados através das entradas / saídas GPIO através de programas no Raspberry Pi 3.
6. Avaliar o desempenho do Raspberry Pi 3 com relação ao processamento de sinais digitais.

1.4. Estado da Arte

De acordo com GHON (2012) a estrutura de tratamento da música tem sofrido algumas modificações nos últimos anos, adaptando-se ao meio dominado pelos computadores, o que comprova que a criatividade tecnológica continua a afetar os meios musicais. Essa adaptação só foi possível devido à digitalização da música, isto é, a utilização do avanço tecnológico como meio de beneficiar e inovar o âmbito musical.

Atualmente, nota-se que a tecnologia tem permitido não só o armazenamento da música, mas também tem possibilitado o surgimento de novos estilos musicais. Pode-se citar como exemplo a música eletrônica, que é completamente baseada na manipulação de dados digitais, não se interessando por sons musicais provenientes da mecânica de um instrumento ou voz, mas pela síntese de um som a partir de suas propriedades físicas fundamentais (MENEZES, 1996, p. 31), isto é, ser capaz de criar novos timbres a partir da manipulação digital da frequência de um som.

Se forem analisados os trabalhos apresentados no Simpósio Brasileiro de Computação Musical, pode-se citar alguns temas como; Inteligência Artificial, Hardware para Áudio, Estrutura de Dados Musicais e Representação, Síntese Sonora, Sistema para Análise Musical, entre outros (MILETTO, 2004). Isso evidencia a importância de estudos sobre as possibilidades que a tecnologia pode trazer para diversas áreas do campo musical.

É importante ressaltar que a gama de instrumentos musicais eletrônicos é cada vez maior no mercado. O estudo sobre a captação de um sinal sonoro analógico para sua identificação como uma nota musical não se limita em apenas ser usado para afinação de um instrumento. Essa técnica vem sendo usada como forma de criar novos timbres a partir de sons da natureza. Isto acontece, pois, instrumentos eletrônicos, como por exemplo sintetizadores, permitem acessar e emular um som a partir de dados digitais. Assim, a captação do som e consequentemente a identificação de qual o seu tom e sua altura na escala harmônica, possibilita sua manipulação e mapeamento por todas as frequências desejadas e, consequentemente sua utilização como se fosse emitido por algum instrumento.

1.5. Organização do Documento

Este trabalho se apresenta dividido em 5 capítulos, onde cada um desses capítulos possuem uma subdivisão em tópicos para um melhor entendimento da realização do trabalho.

O primeiro capítulo está destinado às considerações iniciais do projeto, como motivação, problematização e objetivos. Além disso, são feitas uma introdução e uma pequena abordagem do tema tratado ao longo do documento.

O segundo capítulo é voltado para fundamentação teórica, isto é, conceitos teóricos, definições e outras informações que apresentam relevância para o entendimento do projeto.

O terceiro capítulo é responsável por apresentar o desenvolvimento prático do trabalho, onde é possível ver os passos realizados.

O quarto capítulo é destinado à apresentação dos resultados obtidos, onde podem ser encontradas as análises das formas de ondas.

No quinto capítulo estão contidas as conclusões obtidas com a realização do trabalho e também sugestões para projetos futuros.

2. Fundamentação Teórica

2.1. Revisão de Literatura

A música e a ciência estão atreladas desde os primórdios da humanidade. Na Grécia antiga, por volta de VI a.C., surgiu a primeira relação entre música e aritmética ocorrida no ocidente. Pitágoras percebeu que ao se alterar o comprimento de uma corda, era possível alterar também o som emitido por ela ao fazê-la vibrar (ABDOUNUR, 2000, p.5). A flauta grega era constituída de 4 tubos com mesmo diâmetro e medidas proporcionais a 1, 3/4, 2/3 e 1/2, resultando em sons com frequências inversamente proporcionais ao comprimento dos tubos, de 1, 4/3, 3/2, 2, que correspondem às notas Dó, Fá, Sol e Dó da escala atual. Isso permitiu que, posteriormente, fossem feitas novas relações matemáticas e consequentemente fossem descobertas novas notas musicais, formando o que é chamado de escala Diatônica clássica, compostas pelas notas Dó (C), Ré (D), Mi (E), Fá (F), Sol (G), Lá (A), Si (B) e novamente Dó (CARVALHO, 2009, p. 8).

Com isso, passou-se a utilizar como padrão, a chamada escala temperada, compreendida por um total de 13 notas musicais, C, C#/Db, D, D#/Eb, E, F, F#/Gb, G, G#/Ab, A, A#, Bb, B, as notas da escala Diatônica clássica mais as notas com acréscimo de sustenido (#) e bemol (b), completando o que é chamado de oitava (CARVALHO, 2009, p.9). Assim cada nota da escala musical é descrita por uma frequência diferente, com base na oitava a qual está localizada. Isso introduz ao conceito de semiton (SM) e tom (T); o primeiro remete ao menor intervalo entre duas notas musicais da cultura ocidental e o segundo é a soma de dois semitons (BACKES, 2013).

Em 1876, Alexander Graham Bell, obteve o primeiro marco da eletrônica na música. Ele mostrou, através da invenção do telefone, que um som poderia ser convertido em um sinal elétrico e vice-versa. Posteriormente, em 1906, Thaddeus Chaill criou o dinamofone, um instrumento capaz de gerar sons a partir de dinamos, este foi o primeiro instrumento a produzir sons por meios elétricos (MILETTO, 2004). Em 1915, Lee De Forest inventou o oscilador a válvula, onde

era possível gerar frequências a partir de sinais elétricos. Esta invenção foi a base para criação dos instrumentos eletrônicos idealizados nas décadas seguintes e que são usados até hoje.

Mais tarde, com o surgimento dos computadores, Max Mathews, considerado o pai da computação musical, desenvolveu o primeiro programa de computação para a música, chamado Music I, em 1957. Este programa possuía uma única voz com forma de onda triangular e controlava a afinação, intensidade e duração de um som. O Music I foi a alavanca para o desenvolvimento de uma série de programas musicais de diversas categorias, difundindo o uso de microcomputadores e linguagem de programação em prol da música (MILETTO, 2004).

Na década de 60, surgiu o chamado sintetizador modulador Moog, desenvolvido por Robert Moog. Este instrumento, e outros similares, revolucionaram as técnicas de música eletrônica, pois permitiam a realização de gravações e performance nos estúdios sem consumir tantas horas para preparo e edição dos materiais. Além disso, os músicos passaram a utilizar os sintetizadores para criar seus próprios sons, que muitas vezes não podiam ser produzidos por instrumentos convencionais (MOOG, 1987, p.173-181).

Por último, torna-se relevante evidenciar o surgimento da tecnologia MIDI – *Musical Instrument Digital Interface*. No final da década de 70, os fabricantes de sintetizadores começaram a incorporar circuitos digitais em seus instrumentos. Com isso o emaranhado de cordas físicas foram substituídos por painéis de controle menores e softwares especiais, no entanto cada fabricante possuia seu formato. Em 1983, o protocolo MIDI foi estabelecido como o meio de comunicação padrão entre instrumentos musicais eletrônicos e computadores, assim, a ideia do MIDI como um sistema de notação musical preencheu a lacuna do entendimento musical entre o humano e o computador (LEOPOLD, 1987). Isso fez com que fabricantes de software passassem a produzir diversos aplicativos voltados para o âmbito musical.

Após esses grandes marcos, a manipulação de dados digitais e o aumento da capacidade de processamento vem sendo aplicados junto das novas

tecnologias em prol da música, o que consequentemente resulta nas novas aplicações da eletrônica e da computação para esta área, nos dias atuais.

2.2. Raspberry Pi

O Raspberry Pi (RPI) consiste em um computador de baixo custo do tamanho de um cartão de crédito e considerada capacidade de processamento (SJOGELID, 2013, p.7), em que todo *hardware* é integrado em uma única placa (*system on a chip* - SoC). Ele foi desenvolvido no Reino Unido, pela Fundação Raspberry Pi, com o intuito de ser usado por crianças para ensino de programação e conceitos de eletrônica básica, sem fins lucrativos. Os primeiros modelos do dispositivo foram lançados no mercado em fevereiro de 2012 e posteriormente em outubro do mesmo ano, conhecidos por Modelo A e Modelo B, com memória RAM de 256 MB e 512 MB respectivamente (UPTON, 2014, p.15, 16). Desde então, este dispositivo vem sendo aprimorado e utilizado em diversos trabalhos e projetos, tanto nas faculdades, quanto por desenvolvedores da área.

O modelo mais atual existente no mercador é conhecido por Raspberry Pi 3 (Figura 2.1), doravante chamado RPI3. Ele é considerado um dos minicomputadores mais populares e potentes da atualidade, pois possui um processador Broadcom BCM2837, com capacidade de processamento de 1.2GHz e arquitetura ARM Cortex-A53 (Arquitetura de Processador 64-bits).

O Raspberry Pi 3 vem com 40 pinos GPIO's (General Purpose Inputs and Outputs – Entradas e Saídas de Uso Geral), pinos os quais podem ser programados como entrada e ou saída de dados digitais (RASPBERRY, 2012). Isso possibilita a utilização do RPI3 para a manipulação de dados digitais e comunicação com o meio externo, isto é, as possibilidades de suas aplicações não estão limitadas apenas a programação do computador em si, mas também na utilização do mesmo em circuitos eletrônicos, para automação de sistemas simples, leitura de sensores, entre outras diversas aplicações para um sistema embarcado.

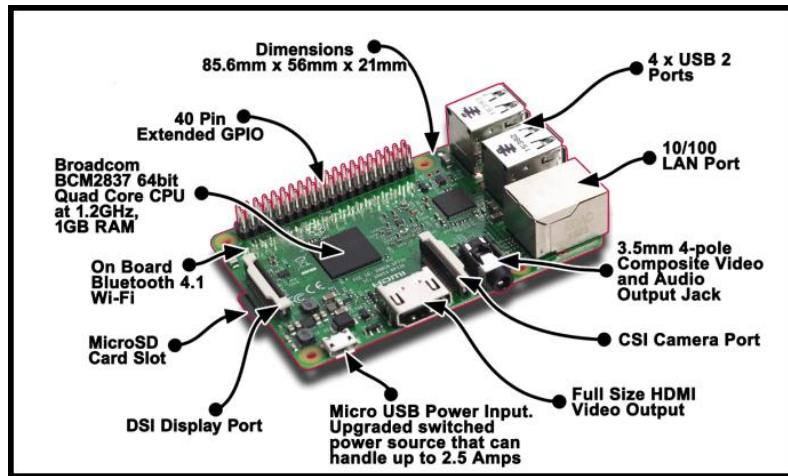


Figura 2.1: RASPBERRY PI 3.

Além disso, o RPI3 possui 4 entradas USB para integração com outros *hardware*s, uma entrada microUSB para alimentação, uma saída *jack P2* para áudio e vídeo, entrada para flat de display e câmera, entrada Ethernet, saída HDMI e um slot para entrada de cartão de memória, o qual é utilizado como HD do RPI3, uma vez que este não possui memória ROM, ou seja, os *softwares* e até o sistema operacional são instalados em um dispositivo de memória externo (RICHADSON; WALLACE, 2013, p.19). A Tabela 2.1 apresenta algumas outras características técnicas deste dispositivo.

RASPBERRY PI 3	
Data de lançamento	29/02/2016
Dimensões	85mm x 56mm x 15mm
Soc	BCM2837
CPU	Quad Cortex A53 @ 1.2 GHz
Processador	ARMv8-A
GPU	400 MHz VideoCore IV
RAM	1 GB SDRAM
Armazenamento	SD / MMC / SDIO
Ethernet	Onboard 10/100 Ethernet RJ45 Jack
Wireless	802.11n / Bluetooth 4.0
Áudio/Vídeo Output	3.5 mm Jack, HDMI Composite
GPIO	40 pinos
Preço	A partir 35 dólares

Tabela 2.1: INFORMAÇÕES TÉCNICAS DO RPI3.

2.2.1. Sistema Operacional

Sistema Operacional é a interface de comunicação entre o usuário e o *hardware*, definindo o ambiente de trabalho e juntamente com outros *softwares*, é utilizado para efetuar configurações no sistema, executar tarefas, e permitir editar, compilar e utilizar diversos aplicativos (SRIRENGAN, 2006).

Pode-se dizer que o sistema operacional atua de forma similar à um gerente executivo, administrando todas as tarefas realizadas por programas e a integração entre os *softwares* e *hardwares* (FLYNN, MCHOES, 2002).

Os sistemas operacionais mais utilizados pelos notebooks e computadores de mesa atualmente são Windows, Linux, OS e Mac, já para o RPI o sistema operacional mais indicado para ser usado é conhecido por Raspbian, um sistema livre baseado em *Debian GNU/Linux*, com mais de 35 mil *softwares* de fácil instalação. Isso se deve ao fato de que, diferente do Windows e do OS, o Linux é sistema de código aberto, isto é, é possível baixar o código fonte de todo o sistema operacional e fazer quaisquer alterações (UPTON, 2014, p.27, 28). Além disso, existem diversos fóruns sobre o Raspbian para auxiliar no uso do Raspberry.

O Raspbian disponibiliza configurações de compilação ajustadas para otimizar a execução de tarefas do Raspberry baseado no seu processamento. Isso permite que ele trabalhe com desempenho significativo evitando travamentos. Além disso, fornece aplicativos apropriados para a arquitetura e as funcionalidades do Raspberry o que faz com que suas aplicações também tenham um ganho no desempenho (RASPBERRY, 2012). Como o RPI não possui memória ROM, também é possível usá-lo com outros sistemas operacionais. Basta instalar cada um deles em dispositivos de memória distintos e trocar estes dispositivos para usar o sistema operacional desejado.

O download e instalação do Raspbian podem ser facilmente encontrados no site oficial da Fundação Raspberry (<https://www.raspberrypi.org>). Os desenvolvedores continuam trabalhando em prol de melhorias do Rasbian, afim de deixá-lo cada vez mais adaptado à capacidade de processamento do RPI e

essas atualizações de sistema podem ser acompanhadas pelo site oficial, mas podem ser feitas também diretas pelo dispositivo.

2.2.2. Portas GPIO (General Purpose Inputs and Outputs)

Como já mencionado, as GPIO's são portas apropriadas para entrada/saída de dados digitais que permitem comunicar o Raspberry com o mundo externo. Essas portas são usadas para realizar a transmissão dos dados entre o dispositivo e o *hardware*, onde podem ser reconhecidos apenas os níveis lógicos 1 (alto) ou 0 (baixo), assim, quando uma tensão de 3.3 V é acionada em um circuito, por exemplo, o *software* que acessa a porta ligada reconhece um recebimento de nível lógico alto (UPTON, 2014, p.226).

Antes de serem implementados projetos no RPI utilizando os acessos da GPIO's é importante reconhecer a funcionalidade de cada entrada, uma vez que esses 40 pinos apresentam funcionalidades particulares. A Figura 2.2 mostra o esquemático com cada uma das funções das GPIO's.

É importante ressaltar que, mesmo com a existência de portas que forneçam uma tensão de alimentação de 5V (Pinos 2 e 4), o funcionamento interno do RPI trabalha com níveis de 3.3V apenas. Isso significa que para implementar circuitos eletrônicos integrados às GPIO's é preciso cuidado, pois se essa tensão de 5V voltar ao RPI pode danificar ou até queimar o *chip*, pois as GPIO's são ligadas diretamente à Broadcom BCM2837, sem qualquer proteção (UPTON, 2014, p.227).

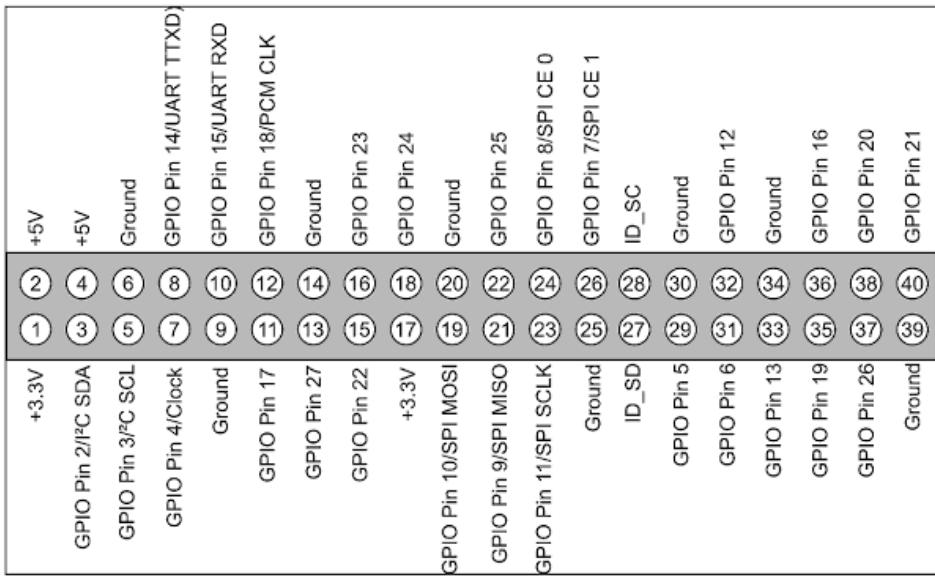


Figura 2.2: FUNÇÕES DA GPIO DO RPI3.

Nota-se pela Figura 2.2 a existência de alguns pinos com características especiais, isto é, pinos que não são simplesmente para saída/entrada digital, alimentação ou aterramento. Os pinos 3 e 5 são nomeados I2C (*Inter-Integrate Circuit* – Circuitos Inter-Integrados). Eles foram projetados para realizar conexões com periféricos de baixa velocidade. O primeiro é usado para dados (*SDA – Serial Data Line*) e o segundo para *clock* (*SCL – Serial Clock Line*). Os pinos 8 e 10 são nomeados por UART (*Universal Asynchronous Receiver/Transmitter*). Se consistem em portas para realizar troca serial de dados binários entre um terminal de dados e um comunicador de dados. Com elas também é possível ajustar a taxa de transmissão de bits (bps). Os pinos 27 e 28 (*ID_SD*) são usados como uma memória que pode ser programada e apagada várias vezes por meio de uma tensão elétrica. Por último, os pinos SPI (*Serial Peripheral Interface* – Interface Periférica Serial), enumerados por 19, 21, 23, 24 e 26, permitem realizar uma comunicação *Full Duplex* síncrona, isto é, o RPI pode se comunicar com algum periférico externo de forma bidirecional (UPTON, 2014).

2.2.3. Programação em Python

Segundo BORGES (2014 p.13), Python é uma linguagem de programação de altíssimo nível (mais próximo da linguagem humana que do código de máquina) orientada a objeto, que possui sintaxe clara e concisa, o que favorece a legibilidade do código fonte. Sua linguagem possui uma tipagem dinâmica e forte, isto significa que no momento em que uma variável for criada por meio de atribuições, o interpretador define um tipo para a variável com as operações que podem ser aplicadas. Assim, o interpretador verifica se as operações são válidas e não faz coerções automáticas entre tipos incompatíveis.

As implementações padrão do Python atualmente compilam, isto é, traduzem instruções do código fonte para um formato intermediário chamado *bytecode* e, posteriormente interpretam as instruções. O *bytecode* é armazenado em disco, para que na próxima execução não seja necessário compilar o programa novamente. Caso uma alteração seja feita no código fonte, o interpretador gera novamente o *bytecode* automaticamente. Isso faz com que o tempo de execução seja reduzido, o que torna a linguagem mais produtiva (BORGES, 2014, p.16).

O Python é um *software* gratuito, que possui código fonte aberto, compatível com diversos sistemas operacionais como, Windows, Linux, OS, entre outros. Além disso, o conjunto de módulos de bibliotecas padrão que acompanham o Python também são implementados com o máximo de portabilidade possível entre as plataformas (LUTZ; ASCHER, 2007, p.37).

2.3. Análise de Sinais

De acordo com HAYKIN; VEEM (2001 p.22), um sinal pode ser formalmente definido como uma função de uma ou mais variáveis, a qual carrega informações sobre a natureza de um fenômeno físico.

Para a geração, extração de informação ou recepção de cada sinal há sempre um sistema envolvido, isto é, uma entidade capaz de manipular um ou mais sinais para realizar uma função. Por exemplo, quando alguém fala, o sistema para emissão da voz são as cordas vocais, e o ouvido é o sistema para

recepção do sinal (HAYKIN; VEEM, 2001 p.23). Assim, pode-se dizer que um sistema é capaz alterar as características de um sinal, caso desejado.

Existem diversas classes de sinais. A partir dessas classes pode-se caracterizar cada sinal e partir de sua característica trabalhar com ele. Para este trabalho é relevante saber a diferença entre as seguintes classes de sinais:

- i. Sinais contínuos e discretos no tempo
- ii. Sinais analógicos e digitais
- iii. Sinais periódicos e não periódicos

2.3.1. Sinais Contínuos e Discretos no Tempo

Pode-se dizer que um sinal contínuo no tempo é um sinal a qual sua característica física varia continuamente no tempo, isto é, possui um valor específico para todo instante de tempo. Já um sinal discretizado no tempo é aquele que pode ter apenas valores pertencentes a um conjunto discreto, em outras palavras, um sinal que possui informação apenas em alguns instantes de tempo (ROBERTS, 2009, p.3). A Figura 2.3 mostra mais claramente a diferença entre estes dois tipos de sinais.

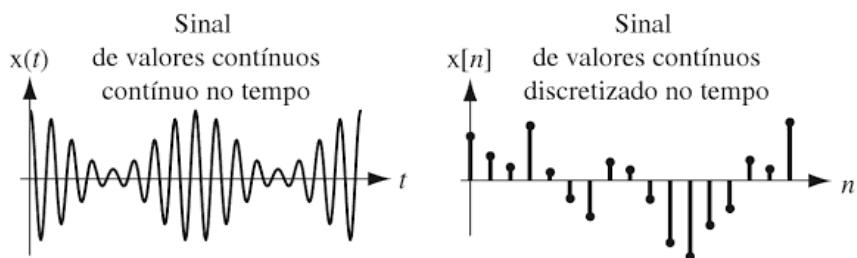


Figura 2.3: SINAL CONTÍNUO X SINAL DISCRETO.

2.3.2. Sinais Analógicos e Sinais Digitais

Geralmente, o conceito de sinal analógico é confundido com sinal contínuo no tempo e o conceito de sinal digital, confundido com sinal discreto. De acordo com LATHI (2006, p.88), os conceitos de continuidade e discretização de sinal estão vinculados à natureza do sinal ao longo do tempo (eixo horizontal). Já os

termos analógico e digital qualificam a natureza da amplitude do sinal (eixo vertical).

Um sinal analógico é definido como um sinal que tem variação no tempo análoga (proporcional) à algum fenômeno físico, com amplitude podendo assumir infinitos valores. Assim, afirma-se que todo sinal analógico é contínuo no tempo, mas nem todo sinal contínuo no tempo é analógico (ROBERTS, 2009, p.3). Em contrapartida, sinal digital se refere à transmissão de uma sequência de valores, os quais são representados por dígitos de alguma forma codificada, geralmente binário. A Figura 2.4 mostra a diferença de um sinal analógico e um sinal digital.

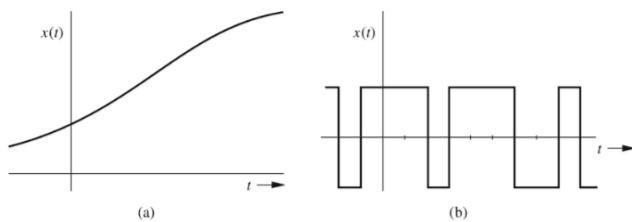


Figura 2.4: (a) SINAL ANALÓGICO; (b) SINAL DIGITAL.

2.3.3. Sinais Periódicos e Não Periódicos

Um sinal é dito periódico quando sua função satisfaz, para todo t , a seguinte condição

$$x(t) = x(t + T) \quad (2.1)$$

onde T é uma constante positiva. Uma vez que esta condição é satisfeita para $T = T_0$, ela também satisfaz $T = 2T_0, 3T_0, 4T_0, \dots$. Logo, o menor valor de T que satisfaz a equação (2.1) é chamado período fundamental de $x(t)$, que define a duração de um ciclo periódico de $x(t)$. O recíproco do período fundamental T é a chamada frequência fundamental deste sinal periódico, que é medida em hertz (Hz) e define o quanto frequentemente $x(t)$ se repete ao longo do tempo (HAYKIN; VEEM, 2001, p.36). Contudo, qualquer sinal que não possua algum valor para T que satisfaça a condição da equação (2.1) é chamado aperiódico ou não periódico.

Uma propriedade importante de um sinal periódico é que $x(t)$ pode ser gerado pela extensão periódica de qualquer seguimento de $x(t)$, com duração equivalente ao seu período fundamental (T_0). Isso se deve ao fato de que os sinais periódicos permanecem inalterados, quanto a sua frequência, ao longo do tempo e por definição deve começar quando $t = -\infty$ (LATHI, 2006, p.88).

2.3.4. Ruído

Um fator relevante a ser abordado na análise de sinais é o ruído. Segundo ROBERTS (2009, p.1), ruído é um sinal, na medida em que é um fenômeno físico variante no tempo, ou seja, é uma interferência da natureza em um sinal que acontece de forma aleatória. Porém, o ruído não apresenta informação útil sobre a característica do sinal e geralmente é considerado indesejável. Ele pode ser considerado uma “sujeira” no sinal e normalmente é amenizado por alguns métodos de tratamento de sinal.

2.3.5. Sistemas digitais e Sistemas Analógicos

Um sistema digital consiste em um dispositivo projetado para trabalhar com informações lógicas ou ainda, quantidades físicas codificadas em uma forma digital. Normalmente, esses dispositivos são eletrônicos, o que não impede que eles possam ser também mecânicos, magnéticos e pneumáticos. Pode-se citar como exemplos desses sistemas, equipamentos de áudio e vídeo, telefone e principalmente os computadores (TOCCI; WIDMER; MOSS, 2007, p.3).

TOCCI (2007, p.3), classifica ainda um sistema analógico como sendo um dispositivo capaz de manipular grandezas físicas, as quais são representadas de forma analógica. Em um sistema analógico, as quantidades físicas podem variar sobre um intervalo de valores. Cita-se como exemplo desses sistemas, amplificadores de áudio, sinal de saída de um receptor de rádio ou um simples interruptor do tipo *dimmer*.

2.4. Processamento Digital de Sinais

O processamento digital de sinais consiste no estudo de regras acerca do comportamento de sinais com funções de variáveis discretas, ou ainda, aspectos envolvidos com sinais que são funções de variáveis contínuas, por meio de técnicas digitais. Também abrange uma análise sobre os sistemas digitais usados no processamento de um sinal (DINIZ, 2014, p.1).

Naturalmente, um sinal tem característica analógica, e carrega consigo alguma informação física. O processamento digital permite realizar a reconstrução desse sinal analógico por meio de uma sequência de valores discretos no tempo, ou seja, cria-se um sinal digital contendo relativamente as mesmas informações carregadas pelo sinal analógico analisado. Isto possibilita trabalhar acerca do sinal com maior facilidade, visto que as informações antes, contendo características físicas, agora são transformadas em dados digitais podendo ser interpretadas por *hardwares* e *softwares*. Assim, o processamento digital de sinais se preocupa com a análise matemática e com a prática do tratamento de informações contidas no sinal como compreendida pelos processadores (NALON, 2000, p.1).

A abordagem do processamento digital recorre a computações numéricas para realizar suas operações. HAYKIN; VEEM (2001, p.33), afirma que uma abordagem analógica do sinal tem garantia de operações tempo real, pois os mecanismos subjetivos às operações em um sistema analógico são todos físicos por natureza, o que não pode ser garantido em um processamento digital. No entanto, o processamento digital apresenta duas principais vantagens em relação ao sistema analógico:

- ✓ Flexibilidade, isto é, a mesma máquina digital (*hardware*) pode ser usada para implementação de diferentes operações de processamento de sinal, simplesmente fazendo alterações em sua programação (*software*), enquanto uma máquina analógica é projetada pra tarefas específicas.
- ✓ Repetitividade, ou seja, uma operação de processamento digital pode ser repetida várias vezes, obtendo a mesma resposta. Um sistema

analógico sofre pequenas alterações, como por exemplo, temperatura do ambiente, o que muitas vezes pode influenciar na resposta.

2.4.1. Teorema da Amostragem e Nyquist

De acordo com LATHI (2006, p.678), amostrar um sinal significa capturar seus valores apenas em instantes discretos no tempo. Isso significa processar um sinal, geralmente analógico, de forma pontual, onde os intervalos de tempo entre um ponto de amostra e outro são constantes. Para isso, é importante manter uma taxa de amostragem do sinal suficientemente alta para permitir sua reconstrução sem erro, ou melhor, com erro dentro de uma dada tolerância.

A taxa mínima em que um sinal pode ser amostrado, enquanto retêm informações suficientes para a reconstrução do mesmo, depende do quão rápido esse sinal varia no tempo, ou seja, a taxa de amostragem pode ser dada como uma função da frequência do sinal (ROBERTS, 2009, p.536). Para facilitar o entendimento, a Figura 2.5 mostra um sinal contínuo no tempo $x(t)$ amostrado com diferentes intervalos de tempo. Na imagem, é possível notar que quanto menor o intervalo de tempo entre as amostras, mais evidentes são as características do sinal que está sendo amostrado.

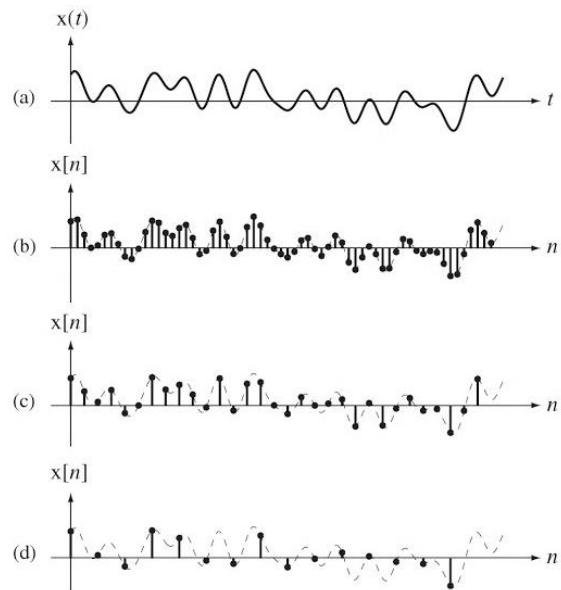


Figura 2.5: (a) SINAL CONTÍNUO NO TEMPO $x(t)$; (b, c, d) AMOSTRAGENS COM n AMOSTRAS DISCRETAS NO TEMPO DO SINAL $x(t)$.

O princípio da frequência de amostragem para reconstrução de um sinal contínuo no tempo é baseado na taxa de Nyquist, ou também chamado, Teorema da Amostragem. Assim, uma vez que um sinal real (análogo) possui um espectro limitado em uma faixa de B Hz, pode-se reconstruir-lo sem erros, a partir de suas amostras, se, e somente se, este for amostrado a uma taxa f_s (frequência de amostragem) maior ou igual a $2B$. Em outras palavras, a taxa de Nyquist ($2B$ Hz) é a menor taxa de amostragem necessária para preservar a informação de um sinal analógico durante sua reconstrução (LATHI, 2006, p.680). A equação (2.2) mostra a relação da frequência de amostragem (f_s) e a equação (2.3) mostra a relação do período de amostragem (T_s) em função da frequência do sinal (B).

$$f_s \geq 2B \text{ Hz} \quad (2.2)$$

$$T \leq \frac{1}{2B} \quad (2.3)$$

2.4.2. *Aliasing*

A taxa de Nyquist é o fundamento para o processamento digital e reconstrução de sinais, pois ameniza o efeito de *aliasing*, muitas vezes chamado de falseamento de sinal. Dessa forma, desde que f_s seja igual ou duas vezes maior do que a largura de faixa B do sinal (em Hertz), o sinal discretizado será constituído de repetições não sobrepostas do sinal original (LATHI, 2006, p.680). Por exemplo, se uma senóide com frequência de 6Hz for amostrada com um intervalo de tempo equivalente a 0.2 segundos, isto é, $f_s = 5\text{Hz}$, o sinal amostrado poderá ter a aparência de um senóide de 1Hz. A Figura 2.6 mostra esse efeito mais claramente.

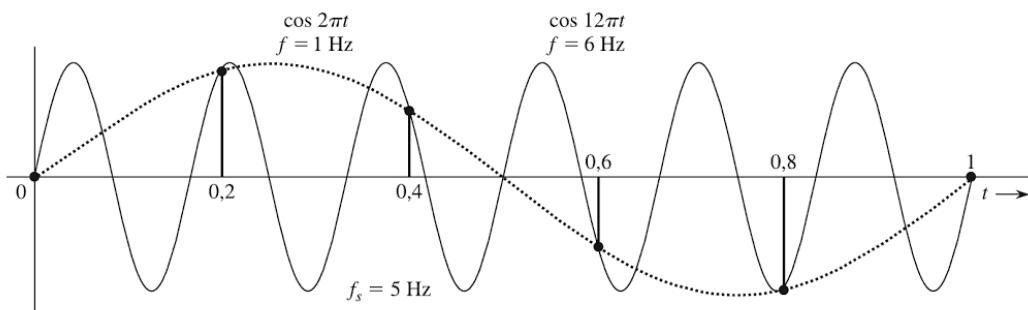


Figura 2.6: EFEITO DO ALIASING EM UM SINAL.

Pode-se notar no exemplo anterior que, se for adotada a mesma frequência de amostragem para amostrar o sinal de 1Hz, o procedimento estaria de acordo com os critérios da taxa de Nyquist, no entanto o falseamento ocorreria da mesma forma. LATHI, (2006, p.693), afirma que uma taxa de amostragem pode ser adequada para uma senóide de frequência mais baixa e obviamente inadequada para uma senóide de frequência mais alta. Isso remete ao fato de que a frequência do sinal real é sempre a menor possível dentro da faixa $|f| \leq f_s/2$.

O *aliasing* acontece, pois entre os intervalos de amostra não há informação a respeito do comportamento do sinal. Assim sendo, para determinar como um sinal se comporta entre as amostras, é preciso especificar restrições adicionais para o sinal de tempo contínuo. Geralmente, na prática, esse conjunto de restrições consiste em exigir que o sinal faça transcrições suaves de uma amostra para outra, uma vez que a suavidade com a qual o sinal no domínio do tempo muda, está diretamente relacionada com que a frequência máxima presente no sinal (HAYKIN; VEEM, 2001, p.292).

Muitas vezes na prática, quando se deseja uma taxa de amostragem reduzida e também uma forma de eliminar o *aliasing* de um sinal, utiliza-se a implementação de um filtro *anti-aliasing*. Ele consiste em um filtro passa-baixa que é colocado antes da discretização do sinal real. Esse filtro deixa passar componentes de frequência abaixo de $f_s/2$ sem distorção, e suprime quaisquer componentes de frequência acima desse valor (HAYKIN; VEEM, 2001, p.293). A relevância de um filtro passa-baixa em um sistema será discutida de forma mais detalhada posteriormente neste documento.

2.4.3. Reconstrução de Sinal (Retentor de Ordem Zero)

O Teorema da Amostragem, diz respeito a quanto rápido a coleta de informações de um sinal analógico deve ser feita para que este possa ser representado de forma única. Porém, ainda é necessário efetuar a reconstrução deste sinal a partir dos dados obtidos. De forma ideal, o processo de reconstrução consiste na substituição de cada amostra por uma função *sinc* (seno cardinal), isto é, o seno de πx dividido por x , centrada no instante de tempo

da amostra e redimensionada pelo valor da amostra $x(nT_s)$ vezes $2f_c / f_s$ e somando todas as funções criadas (ROBERTS, 2009, p.543).

Como pode ser esperado, este método de reconstrução não é utilizado na prática, visto que a sua função não é implementável. Essa não implementação se deve a dois fatores. Primeiro, a função representa um sistema não casual, ou seja, a saída $x(t)$ depende de valores passados e futuros da entrada $x[n]$. Segundo, a influência de cada amostra se estende a um intervalo infinito de tempo (HAYKIN; VEEM, 2001, p.295).

Na prática, a técnica mais usual para a reconstrução de um sinal é chamada de retentor de ordem zero (*Zero Order Hold - ZOH*). Assim, o sinal produzido por essa técnica tem a forma em degrau de escada que acompanha o sinal original (ROBERTS, 2009, p.544). O ZOH é representado matematicamente como uma soma ponderada de pulsos retangulares, deslocados em múltiplos inteiros do intervalo de amostragem. Desse modo:

$$h(t) = \begin{cases} 1, & 0 < t < T_s \\ 0, & \text{do contrário} \end{cases} = \text{ret}((t - T_s/2)/T_s) \quad (2.5)$$

Essa técnica torna-se então, muito útil, uma vez que as amostras na forma de código numérico podem ser um sinal de entrada para um conversor D/A ou saída de um conversor A/D, componentes que geralmente são implementados em circuitos eletrônicos como pontes entre o mundo digital e analógico e vice-versa. A Figura 2.7 mostra um exemplo da forma de onda gerada a partir da técnica ZOH.

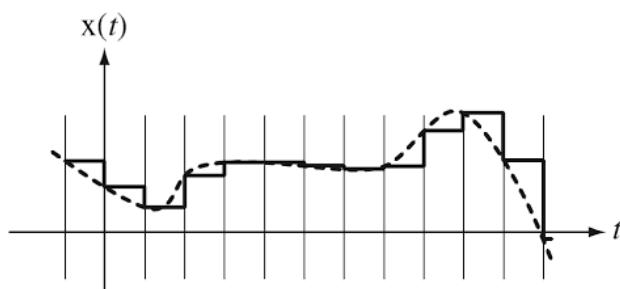


Figura 2.7: FORMA DE ONDA GERADA PELO ZOH.

Por fim, é importante ressaltar que o retentor de ordem zero provoca, inevitavelmente um atraso em relação ao sinal de original, isso acontece pois ele é regido por uma função casual, ou seja, sua saída não depende de nenhum valor de entrada, ela depende apenas do valor da próxima amostra.

2.5. Eletrônica de Sistemas

2.5.1. Transdutores

Geralmente, uma variável física não é necessariamente elétrica, por exemplo, variáveis como temperatura, pressão, umidade, velocidade, etc. não possuem, na maioria das vezes, nenhuma relação com uma tensão ou corrente elétrica. Um transdutor pode ser classificado como um dispositivo capaz de converter uma variável física em uma variável elétrica. Pode-se citar como exemplo de um transdutor, microfones, fotocélulas, medidores de fluxo, tacômetro, entre outros (TOCCI; WIDMER; MOSS, 2007, p.377). Assim, basicamente, um transdutor tem uma saída elétrica representada por uma tensão ou corrente analógica proporcional à variável física que está sendo monitorada.

2.5.1.1. Microfone

PATSKO (2006, p.24), define microfone como um captador de som, destinado a converter vibrações sonoras em sinais elétricos. Existe uma variada gama de microfone, com funcionamento baseado em diversos princípios. O mais comum deles é o microfone de eletreto, mostrado pela Figura 2.8. Este tipo de microfone é usado em aparelhos telefônicos, computadores e até em gravação de áudio.



Figura 2.8: MICROFONE DE ELETRETO.

A maioria dos microfones utiliza-se de um diafragma para a captação, isto é, uma película fina e flexível que vibra na presença de ondas sonoras. Em um microfone de eletreto, além do diafragma, há uma pequena placa de metal, onde juntos eles compõem um capacitor. Dessa forma, quando o diafragma vibra, a sua distância em relação à placa de metal varia e, consequentemente, varia-se a capacidade desse conjunto. Isso resulta na variação da tensão existente entre a película e a placa, reproduzindo um sinal elétrico análogo à vibração do diafragma. Como o sinal é alternado, a tensão varia acima e abaixo de uma tensão de referência (padrão 0V). Além disso, o microfone de eletreto possui internamente um FET (*Field Effect Transistor* – Transistor de Efeito de Campo), que tem a função de atuar como um *buffer*, eliminando problemas de impedância e capacidade que podem ocorrer durante a conexão deste componente com o destino final do sinal (PATSKO, 2006, p.25). Geralmente este destino final é um conversor A/D, o qual não reconhece sinais de tensão negativos. Nestes casos, são estabelecidos uma tensão de referência maior que 0V, para que durante a variação do sinal não haja perda de informação. A Figura 2.9 ilustra o interior de um microfone de eletreto para facilitar o entendimento do seu funcionamento.

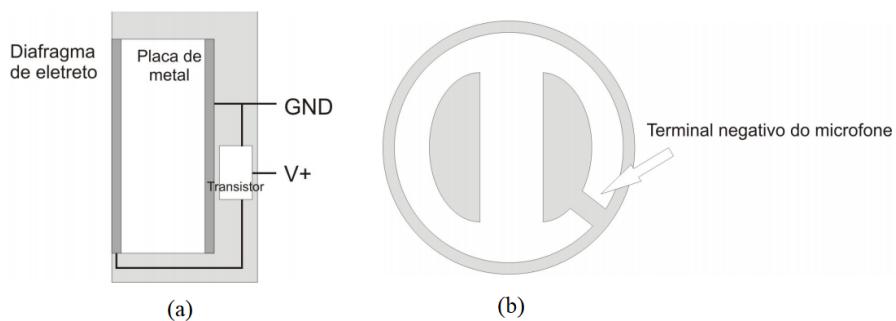


Figura 2.9: INTERIOR DE UM MICROFONE DE ELETRETO; (a) VISTA LATERAL; (b) VISTA FRONTAL.

2.5.2. Conversor Analógico-Digital (A/D)

Um conversor A/D é um circuito integrado que recebe uma tensão analógica na sua entrada e depois de um certo tempo, produz um código digital de saída similar à sua entrada (TOCCI; WIDMER; MOSS, 2007, p.377). Estes dados binários permitem que o sinal seja compreendido por processadores, ou microcontroladores.

A temporização de um conversor A/D é dada a partir de um sinal de *clock* de entrada, normalmente fornecido pelo processador. Desse modo, a unidade de controle gera uma sequência de operações apropriadas, por meio de seus circuitos lógicos, em resposta ao comando START, que inicia o processo de conversão (TOCCI; WIDMER; MOSS, 2007, p.377). TOCCI (2007) ainda, classifica a eficiência destes dispositivos de acordo com a quantidade de bits para o qual eles foram projetados, isto é, a quantidade de amostras coletadas de um sinal a cada segundo.

2.5.3. Filtros

De forma geral, o funcionamento de um filtro consiste em permitir a passagem de um determinado intervalo de frequência enquanto rejeita outra. Eles são usados para separar sinais indesejados, eliminar ruídos, melhorar e modificar a qualidade de um sinal. Os filtros podem ser separados em passivos e ativos. Um filtro passivo é construído com resistores, capacitores e indutores. Geralmente são usados para situações acima de 1MHz, não possuem ganho de potência e são relativamente difíceis de serem sintonizados. Em contrapartida, um filtro ativo é constituído resistores, capacitores e amplificadores operacionais. Eles são úteis abaixo de 1MHz, têm ganho de potência e podem ser facilmente sintonizados (BATES; MALVINO, 2007, p.221).

Todo filtro possui um ganho G de tensão, dado em decibéis (dB), calculado pela equação (2.6). Idealmente esse ganho é unitário na faixa de passagem e zero na faixa rejeitada, o que acarreta na transição repentina do sinal na frequência de corte (ω_c). Isso não acontece na prática, assim, é necessário

estabelecer um ganho mínimo para a faixa de passagem (f_p) e um ganho máximo para a faixa de bloqueio (f_b) em torno de ω_c (LATHI, 2006, p.407).

$$A(dB) = 20 \log V \quad (2.6)$$

Os filtros são descritos matematicamente por sua função transferência, isto é, a razão entre o valor de tensão de saída e o valor de tensão de entrada, o qual é mostrado graficamente pelo ganho de tensão versus a frequência. Dentre os cinco tipos de filtros existentes, destacam-se os filtros passa-baixa, passa-alta e passa-faixa. A Figura 2.10 mostra os gráficos ideais e reais para estes tipos de filtro.

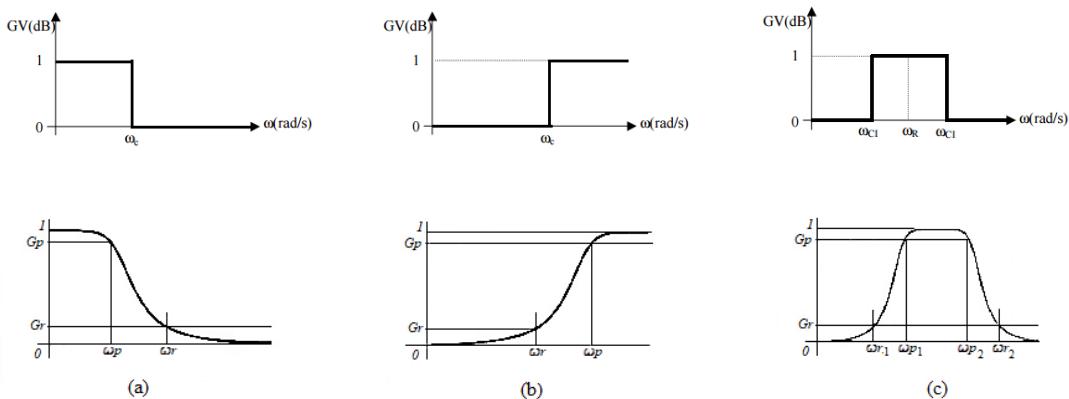


Figura 2.10: (a)PASSA-BAIXA; (b)PASSA-ALTA; (c)PASSA-FAIXA.

2.5.3.1. Filtro Passa-Baixa

O filtro passa-baixa tem a função eliminar altas frequências, deixando passar as frequências menores mais baixas que a frequência de corte (ω_c). Ele é representado de forma mais elementar por um circuito RC, ou seja, um circuito contendo um capacitor e um resistor, como mostra a Figura 2.11 (ROBERTS, 2009, p.447).

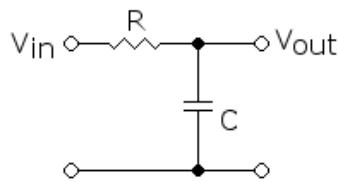


Figura 2.11: CIRCUITO RC - FILTRO PASSA-BAIXA.

ROBERTS (2009, p.448) explica que, em baixas frequências, a impedância do capacitor é muito maior, em magnitude, do que a impedância do resistor, o que faz com que V_{in} e V_{out} tenham praticamente o mesmo valor de tensão. Quando o circuito é submetido a altas frequências, a impedância do capacitor torna-se muito menor que a do resistor, fazendo com que V_{out} tenda a zero.

2.5.3.2. Filtro Passa-Alta

O filtro passa-alta, opera de maneira contraposta ao filtro passa-baixa. Desse modo, um filtro passa-alta elimina frequências abaixo da frequência de corte e permite a passagem das frequências superiores. Seu circuito eletrônico também consiste em um capacitor e um resistor, no entanto, diferente do filtro passa-baixa, neste filtro o sinal de saída é observado nos terminais do resistor (BATES; MALVINO, 2007, p.223). O circuito eletrônico básico que compõe é mostrado pela Figura 2.12.

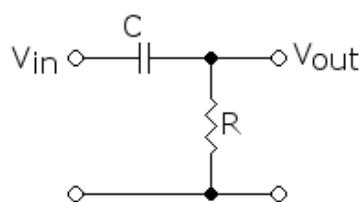


Figura 2.12: CIRCUITO RC - FILTRO PASSA-ALTA.

2.5.3.3. Filtro Passa-Faixa

Os filtros passa-faixa, são como uma junção do passa-alta com o passa-baixa, ou seja, ele estabelece uma faixa entre duas frequências para que possa

ser passado o sinal. Sua forma mais básica é constituída por um circuito RLC, isto é, um circuito com um resistor um capacitor e um indutor, ver Figura 2.13. Em frequências muito baixas o capacitor se comporta como um circuito aberto, já para altas frequências o indutor se comporta dessa forma.

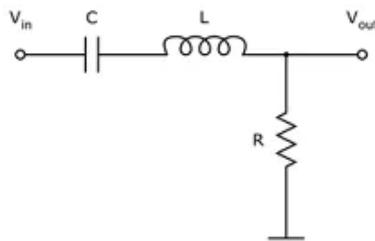


Figura 2.13: CIRCUITO RLC - PASSA-FAIXA.

2.6. Fundamentação Musical

2.6.1. Notas Musicais

Nota musical é nome dado a um dos elementos mais básicos de um som, é a nomenclatura dada a este de acordo com sua frequência fundamental e, assim, tornar-se possível mapeá-lo de acordo com sua altura, ou seja, quanto maior frequência mais alto na escala a nota está (CHEDIAK, 1986). Sua alocação ocorre de acordo com a classe e a oitava em que se encontra sua frequência.

Como já mencionado, oitava é o conjunto de notas, onde a primeira e a última possui relação de frequência 1 para 2. Já a classe da nota pode ser definida como a localização da mesma dentro de uma presente oitava. Isto remete ao fato da relação entre a música e física de um som matematicamente, uma vez que uma oitava apresenta relação de 1 para 2 na frequência de uma mesma nota, além de ser dividida entre 12 notas distintas, infere-se matematicamente que aumentar um semitom, ou ainda, o menor intervalo entre duas notas distintas, corresponde a aumentar $2^{(n/12)}$ vezes a frequência de uma nota, onde n é o número de semitonos aumentados (CARVALHO, 2009). Isso torna possível o mapeamento da frequência de todas as notas existente em uma escala.

Com base nos conceitos de tom (T) e semiton (ST) são feitas as formações de escalas tônicas, onde as duas principais são a escala maior, na sequência de T, T, ST, T, T, T, ST a partir da nota fundamental, e a escala menor natural na sequenciada por T, ST, T, T, ST, T, T. Isso infere na formação dos acordes maiores e menores. Um acorde se consiste na junção de três ou mais notas distintas (ADOLFO, 1989). Os acordes são construídos a partir da frequência da nota fundamental da escala, e podem ser descritos por f , $4/5f$ e $2/3f$, como os acordes menores (possui um 'm' ao lado do acorde) e f , $5/4f$, $3/2f$ como acorde maior.

A definição de notas e acordas introduz um importante conceito para identificação de um som, chamado polifonia musical. Um som musical pode ser classificado como monofônico ou polifônico. O primeiro pode ser entendido como um único instrumento tocando apenas uma nota, que não é necessariamente monótona, isto é, uma senóide pura. Já polifonia se consiste na junção de várias notas ou instrumentos (OGASAWARA, 2008). O fato é que para identificar um som polifônico não basta apenas reconhecer frequências, como no caso monofônico, pois como se trata de mais de uma nota sendo tocada ao mesmo tempo a frequência de uma nota gera interferência em outra. Para este caso, são necessários estudos mais avançados de métodos computacionais capazes de lidar com essas interferências.

2.6.2. Percepção de um Som

Os sons musicais e os ruídos podem ser facilmente distinguidos. O que caracteriza fisicamente sons musicais é a grande coerência de frequências harmônicas, ou seja, múltiplas umas das outras, presentes nestes sons. Isto ocorre, por exemplo, ao fazer vibrar uma corda esticada entre dois pontos, ou o ar contido em um tubo, o que levou à construção dos instrumentos musicais primitivos. Assim, entende-se música como um sequenciamento de combinações de sons com essas características (CARVALHO, 2009). Já o ruído é a mistura de sons ou tons cujas frequências diferem entre si por um valor inferior ao poder de discriminação de frequência do ouvido, ou seja, é qualquer sensação sonora considerada indesejável.

A forma mais elementar de uma onda sonora é a senoidal, portanto, uma onda periódica (ZUBEN, 2004, p.14). A Figura 2.14 mostra um exemplo com uma senóide equivalente a 400Hz, com seus respectivos parâmetros de identificação; 'A' representa a amplitude dessa onda e 'T' o período da mesma. Assim, uma onda senoidal pode ser descrita matematicamente, a partir dos seus parâmetros, conforme a equação (2.7).

$$x(T) = A \cdot \text{sen}(2\pi f \cdot T) \quad (2.7)$$

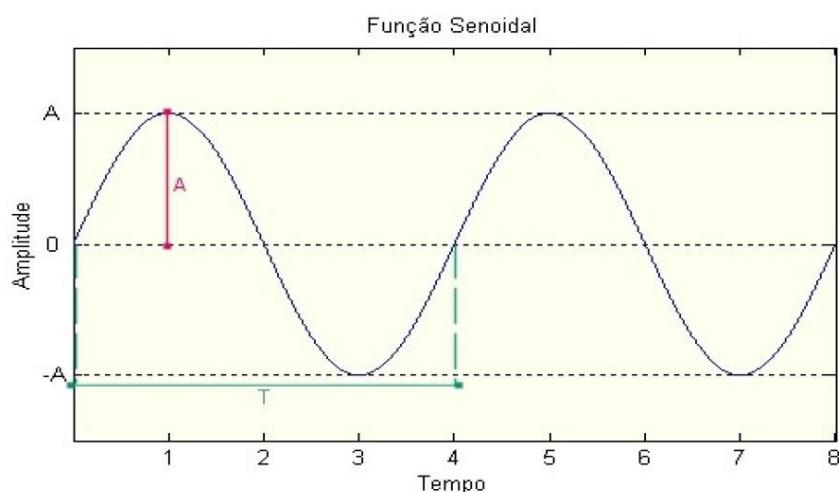


Figura 2.14: FORMA DE ONDA SENOIDAL DE 400Hz.

Estes são os principais parâmetros responsáveis pela percepção de um som, isto é, parâmetros que definem as características da onda sonora. A amplitude 'A' é diretamente relacionada a intensidade do som, ou seja, quanto mais alto o som maior a amplitude de sua onda. O período 'T' está relacionado à frequência, pois define o quão grave ou agudo é um som. Além destes dois aspectos existentes na percepção sonora há um terceiro aspecto, chamado timbre, o qual é responsável pela diversidade sonora. O timbre é o que difere o som produzido por um instrumento de outro (LAZZARINI, 1998). A Figura 2.15 mostra as diferentes formas de ondas quando uma mesma nota é tocada por instrumentos diferentes.

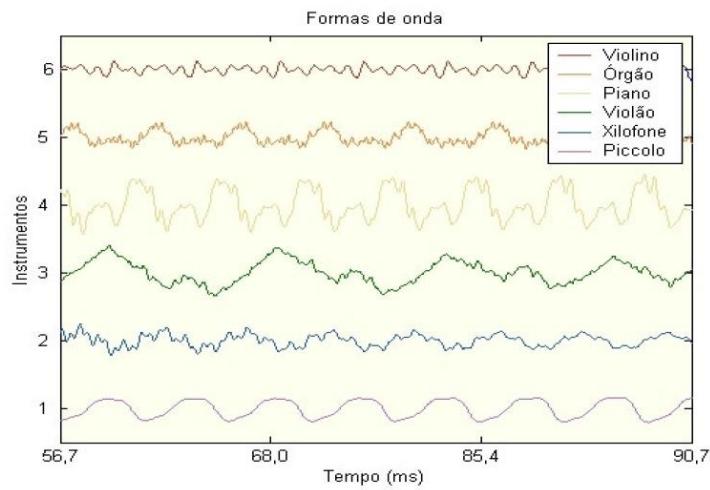


Figura 2.15: FORMAS DE ONDAS DE DIFERENTES INSTRUMENTOS PARA UMA MESMA NOTA MUSICAL.

Com base nos parâmetros de uma onda sonora, é importante ressaltar o comportamento dessa onda ao longo do tempo, quando esta é emitida por um instrumento musical. Esse comportamento é chamado de envelope sonoro ou envoltória sonora. Quando um instrumento musical emite um som, esse som possui basicamente 4 etapas. Elas são chamadas ataque, quando a onda possui maior intensidade, decaimento, quando atingi sua intensidade máxima e começa a decair, sustentação, onde sofre uma leve perda de intensidade, e relaxamento, onde muda do estado de sustentação para o estado de silêncio (MILETTO, 2004). Estes comportamentos podem variar de acordo com o timbre do instrumento. A Figura 2.16 mostra este processo tendo como exemplo o envelope sonoro de uma onda emitida por um Piccolo.



Figura 2.16: ETAPAS DE UMA ONDA SONORA EMITIDA POR UM INSTRUMENTO AO LONGO DO TEMPO.

2.6.3. Parciais Harmônicas

Um som, como já foi dito, é um fenômeno físico que pode ser caracterizado de acordo com sua frequência. Isso remete a possibilidade de relacionar, de forma simples, frequências múltiplas inteiras a partir de uma frequência fundamental (f_0), tal relação é conhecida como parciais harmônicas ou simplesmente harmônicos (OGASAWARA, 2008).

Uma série harmônica é classificada por CHEDIAK (1986, p.42) como uma série de subvibrações geradas a partir de um som principal. Um exemplo pode ser dado ao fazer a corda de um violão vibrar por toda sua extensão, gerando uma determinada frequência. Este mesmo corpo pode vibrar em duas metades, um terço, um quarto de seu comprimento e assim por diante, dando origem a uma série harmônica baseada na frequência da corda solta.

É importante ressaltar que a decomposição de um som em parciais, resulta na obtenção de espectros, os quais são responsáveis pela descrição da onda sonora. Existem três tipos de espectros (ZUBEN, 2004, p.14). A Figura 2.17 mostra como exemplo o espectro de uma nota A4 (440 Hz) emitida por um Oboé, em escala linear.

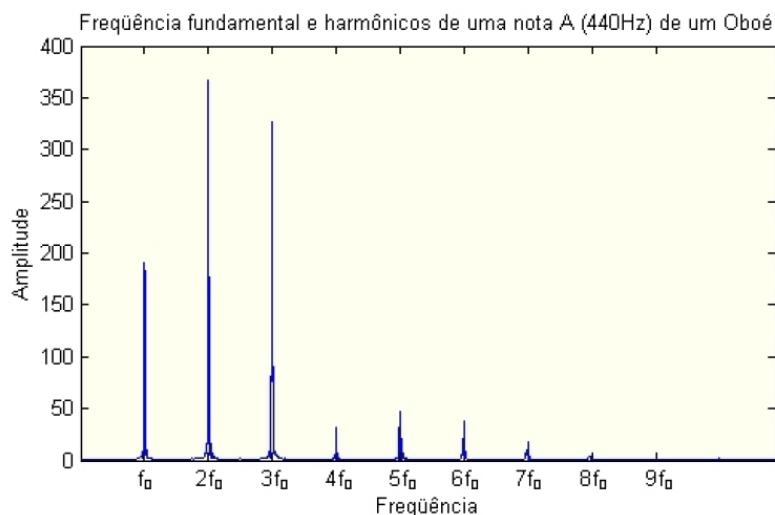


Figura 2.17: ESPECTRO DE FREQUÊNCIAS FUNDAMENTAL E HAMÔNICAS DE UMA NOTA A4.

ZUBEN (2004, p.14) afirma ainda que existem três tipos de espectros. Primeiro cita-se o espectro de parciais harmônicas, o qual as características já

foram abordadas nos parágrafos anteriores. Além deste, existe o espectro de parciais inarmônicos, constituindo frequências parciais com valores de números fracionários a partir de f_0 , por exemplo, uma série com valores equivalentes à 100Hz, 22Hz, 450Hz, etc. Estes tipos de parciais representam os sons de instrumentos de percussão, isto é, baterias, xilofones, sinos entre outros (VILLA-LOBOS, 2007). O outro tipo de parciais espectrais são os ruídos, com frequências sem ligação.

2.6.4. Computação e Música

O ouvido humano é capaz de perceber sons entre uma faixa de frequência de 20 Hz à aproximadamente 20 KHz, no entanto o ouvido é mais sensível para sons com frequências em torno de 1 KHz, isso faz com que algumas pessoas não sejam capazes de ouvir sons acima de 15 KHz. Assim, o ouvido possui um limiar sensível equivalente a 1 Watt acústico de potência e a audição possui um limiar de 1×10^{-12} Watt (ZUBEN, 2004, p.18). Como na música é usada escala logarítmica, esses valores são descritos em *dB* como 0 *dB* (limiar de audição) e 120 *dB* (limiar da sensibilidade).

Tendo por base a capacidade e sensibilidade de um ouvido humano para identificação de sons e também o princípio de amostragem de Nyquist, pode-se dizer que, para amostrar sinais sonoros dentro da faixa de audição humana é necessário no mínimo uma taxa de amostragem de 40 KHz, ou seja, um frequência duas vezes maior que a frequência máxima audível ao ouvido humano. De acordo com MILETTO (2004), um equipamento de áudio profissional é capaz de realizar 44.100 amostras por segundo de um sinal, isto equivale a uma frequência de amostragem equivalente a 44.1 KHz.

MILLETO (2004), classifica ainda que os fatores qualitativos para digitalização de um sinal de áudio são divididos em três, dados:

- 1) O número de amostras por segundo, isto é, a frequência de amostragem do sinal, também chamada por *sample rate* (taxa de amostragem).

- 2) O número de bits em cada amostra, o que representa a resolução de uma amostra. Este fator é dependente do sistema digital utilizado para digitalizar o sinal, assim, quanto maior o número de bits do sistema mais números poderão ser representados na amostra.
- 3) O último fator é o projeto e a qualidade do circuito de áudio utilizado para efetuar a filtragem e amplificação do sinal sonoro.

Em uma gravação de áudio, se o sinal de uma fonte sonora for gravado com o *sample rate* de 48 KHz, será possível captar bandas com uma frequência de até 22 KHz, de modo a proporcionar uma relativa fidelidade nos agudos. Em termos dos bits, se um sinal de áudio for gravado com uma resolução de 8 bits, terá uma forma de onda subdivida em 256 fragmentos. Isto é insuficiente para obter fidelidade no sinal (ALVES, 2006, p.24).

Além desses fatores, para a aplicação de computação na música, também é preciso levar em consideração o conceito de latência. Como já abordado neste documento, o processo de digitalização de um sinal envolve um certo atraso. Este tempo de processamento é conhecido como latência de um sistema digital, ou seja, é o tempo consumido por *buffers*, existentes nos sistemas, para que o sinal percorra entre a entrada e a saída (LAGO, 2004). Normalmente este tempo se faz presente na grandeza de milissegundos (ms) e é fundamental leva-lo em consideração, principalmente em situação de processamento *real-time*.

2.7. *Fast Fourier Transform (FFT's)*

As FFT's também conhecidas como Transformada Rápida de Fourier é uma das técnicas utilizadas para identificação de sinais sonoros conforme suas respectivas frequências. A FFT é definida como um método otimizado da Transformada de Fourier, e possui uma representação numérica mais simples, para que assim possa ser implementada de forma computacional. Como já foi citado neste documento um som pode ser definido por meio de sua frequência (nota), forma de onda (timbre) e amplitude (intensidade). Além disso, é de conhecimento que os sinais estudados neste documento se tratam de sinais periódicos, uma vez que se tratam de sinais sonoros.

Para melhor entender o funcionamento da Transformada de Fourier é necessário introduzir o conceito de impulso, ou também chamada função delta Dirac. A função delta Dirac $\delta(t)$, é definida como a função que satisfaz:

$$\delta(t) = \begin{cases} 0 & \forall t \neq 0 \\ \infty & t = 0 \end{cases} \quad (2.8)$$

$$\int_{0-}^{0+} \delta(t) dt = 1$$

Nota-se então que esta função será zero em todos os pontos exceto quando $t = 0$, logo a função delta Dirac pode ser representada graficamente, deslocada no tempo, conforme mostrado na Figura 2.18.

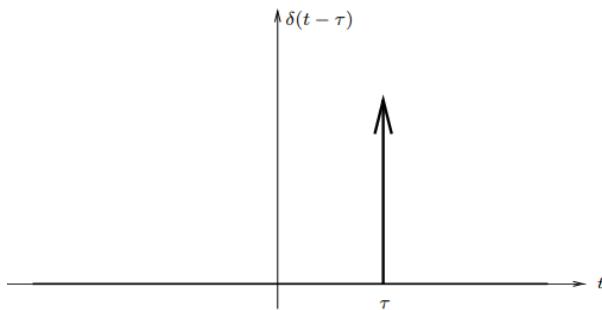


Figura 2.18: FUNÇÃO DELTA DIRAC (IMPULSO) DESLOCADO NO TEMPO.

Assim, de acordo com ROBERTS (2009, p. 433) a resposta ao impulso total de um sistema contendo sistemas associados em paralelos corresponde a soma das respostas aos impulsos individuais. Uma vez que a Transformada de Fourier de uma soma de função no domínio do tempo equivale à soma das Transformadas de Fourier das funções individuais, a resposta em frequência de um sistema contendo sistemas em paralelos é a soma da resposta em frequência de cada sistema individual.

Segundo JOSEPH FOURIER (1768-1830), qualquer função $f(x)$ pode ser escrita na forma de somatória de uma série de funções seno e cosseno da seguinte forma:

$$f(x) = a_0 + a_1 \sin(x) + a_2 \sin(2x) + a_3 \sin(3x) + \dots + b_1 \cos(x) + b_2 \cos(2x) + b_3 \cos(3x) + \dots \quad (2.9)$$

Este processo pode ser melhor entendido através da Figura 2.19, a qual ilustra a transformada de Fourier de um sinal. Deste modo, é possível visualizá-lo no domínio do tempo e sua respectiva transformada no domínio da frequência.

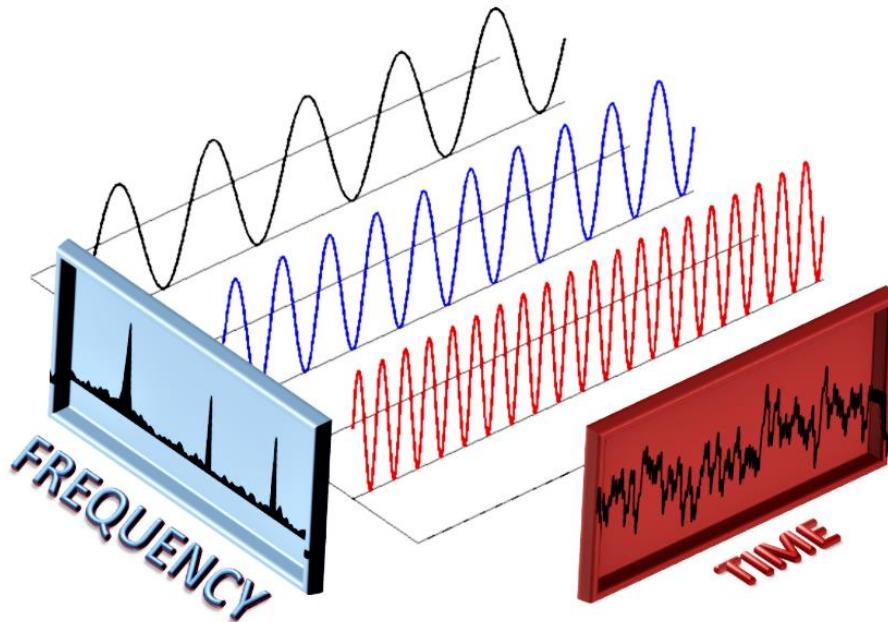


Figura 2.19: REPRESENTAÇÃO DE UM SINAL (TEMPO) E SUA RESPECTIVA TRANSFORMADA DE FOURIER (FREQUÊNCIA).

Assim, pode-se afirmar que a Transformada de Fourier decompõe um sinal em suas componentes elementares seno e cosseno. Isto é, a transformada de Fourier é definida como um mapeamento que leva um sinal a uma distribuição complexa (DA PENHA, 1999, p.16). Como é sabido, uma distribuição complexa possui, por exemplo $X(\omega)$, é decomposta em representação polar, ou seja, possui “raio” e “angulo”.

$$X(\omega) = \rho(\omega)e^{i\theta(\omega)} \quad (2.10)$$

A partir dessa informação é possível, através de deduções matemáticas, interpretar as funções $\rho(\omega)$ e $\theta(\omega)$ separadamente da seguinte forma:

- $\rho(\omega)$: A distribuição a qual possui informações acerca das cossenóides existentes em um sinal $x(t)$, e as amplitudes nelas presente.
- $\Theta(\omega)$: A distribuição que informa o defasamento relativo entre as cossenóides q compõem o sinal $x(t)$.

Com base neste aspecto, é dito que a transformada de Fourier pode ser descrita matematicamente, na sua forma discreta, conforme a equação abaixo:

$$F(\omega) = \sum_{x=0}^{N-1} f(x)e^{-i\omega x} \quad (2.11)$$

onde N se consiste no número de pontos amostrados e ω representa a frequência.

De acordo com KAWAKAMI (2001, No. 6), uma das limitações encontradas na transformada de Fourier é que ela não possibilita a análise de diferentes partes do sinal, ou seja, quando um trecho possui muitos ruídos ou não apresenta informações de interesse da análise, este trecho não pode ser desconsiderado, acarretando assim um comprometimento no processamento do sinal.

A partir da existência deste problema, é introduzido o conceito de transformada janelada de Fourier, utilizado como base para desenvolvimento computacional da *FFT*. Este método por sua vez dividi o sinal em certa amostrar janeladas, afim de selecionar as regiões mais importantes a serem analisadas pelo sinal. Esta pode ser representada pela Equação 2.12.

$$F(p, \omega) = \sum_{x=0}^{N-1} f(x)w(x - p)e^{-i\omega x} \quad (2.12)$$

onde $w(x)$ representa uma função janelada, responsável por limitar o trecho a ser levado em consideração durante a análise de um sinal, e p diz a posição desta janela “dentro” do sinal (KAWAKAMI, 2001, No. 6).

A Figura 2.20 mostra um exemplo de aplicação deste método. Como pode ser visto, o sinal de teste é dividido em oito janelas distintas para que assim seja possível analisar cada intervalo conforme o desejado.

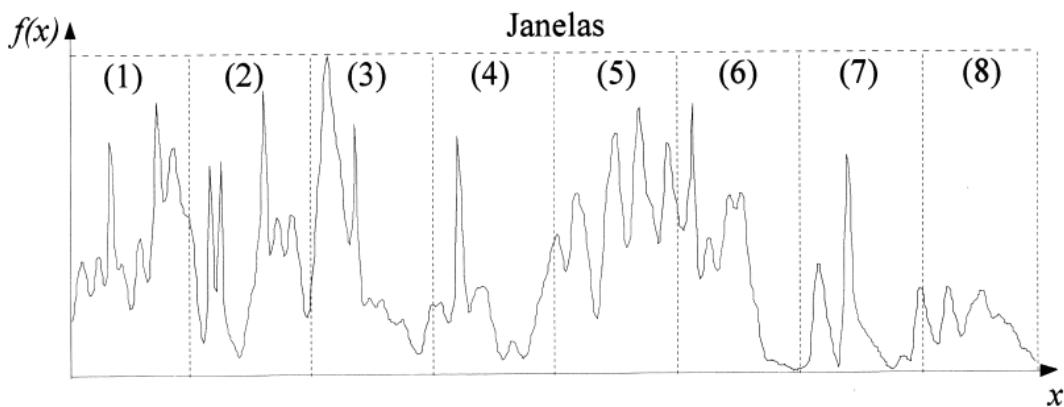


Figura 2.20: PROCESSO DE JANELAMENTO DE UM ESPECTRO DE SINAL.

3. Metodologia

Este capítulo está destinado a apresentar, detalhadamente, o desenvolvimento prático realizado durante este trabalho. Nele podem ser encontrados os modelos dos componentes usados, a montagem dos mesmos, além de justificativas para a escolha de cada um dos elementos do projeto.

3.1. Inicialização do Raspberry PI 3

A compra do modelo RPI3 pode ser feita facilmente pela internet tanto em sites internacionais como em sites nacionais e os preços variam em torno de R\$ 170,00 para ambas as opções. Neste caso, devido ao curto tempo para realização do projeto, o RPI3 foi comprado aqui mesmo no Brasil, afim de evitar atrasos no trabalho. Foi comprado, um kit completo contendo o RPI3, um case, uma fonte de alimentação e dissipadores de calor para o núcleo de processamento.

Durante o prazo de entrega do produto foi realizado um estudo das características e configurações relevantes para utilização do RPI3 em projetos, onde foi possível constatar a necessidade de outros elementos para configurar este minicomputador. Assim, foram utilizados os seguintes componentes para inicializar o RPI3:

- (1) Fonte de Alimentação;
- (2) Dissipadores de Calor;
- (3) Cartão de Memória de 8GB;
- (4) Teclado USB;
- (5) Mouse USB;
- (6) Cabo HDMI;
- (7) Cabo Ethernet *RJ45*;

Então, foi montada a bancada de trabalho, conforme mostrada na Figura 3.1. Além disso, foi utilizado como monitor de trabalho uma televisão comum com entrada HDMI para que fosse possível monitorar o processo.

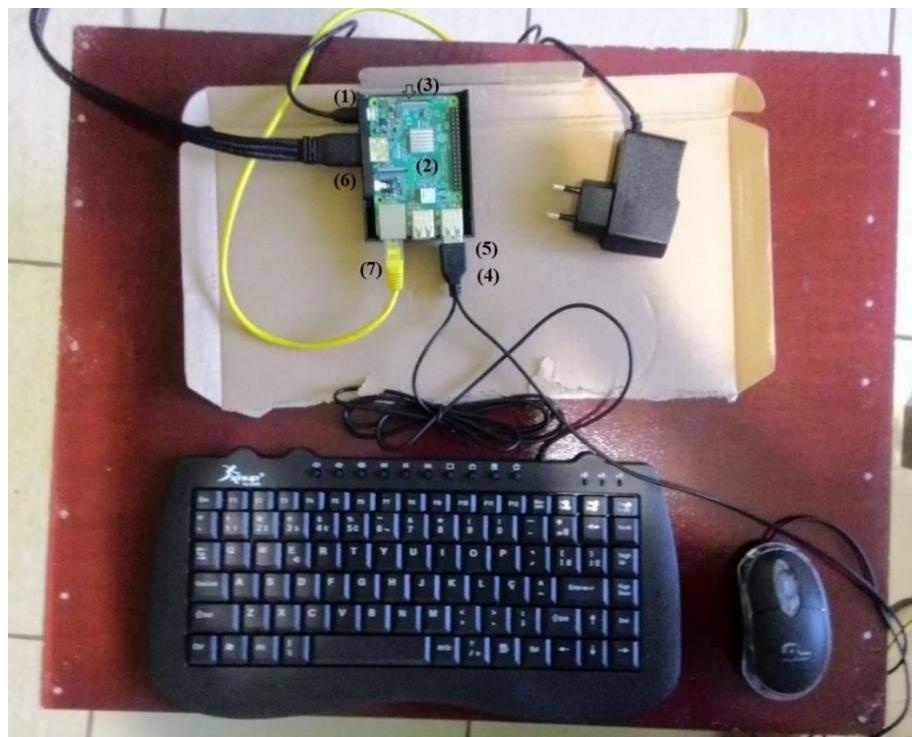


Figura 3.1: BANCADA DE TRABALHO PARA INICIALIZAÇÃO DO RPI3.

3.1.1. Instalação do Sistema Operacional e Configurações Iniciais

Tendo-se elaborado a bancada de trabalho, partiu-se para instalação de softwares e do sistema operacional. Inicialmente foi utilizado um notebook comum a fim de efetuar o download do sistema operacional, isto é, o Raspbian. Para isso, foi feito o download do *New Out Of Box Software* (NOOBS), disponibilizado gratuitamente no endereço downloads.raspberrypi.org/noobs. Como já citado, o RPI3 não possui memória ROM, assim, faz-se necessário o uso de um dispositivo de memória externa com capacidade de armazenamento mínima de 4GB, o qual será usado para instalação do sistema operacional, além dos softwares e arquivos.

Com o intuito de realizar a instalação do sistema operacional no cartão de memória, configurou-se o mesmo em uma formatação específica seguindo os seguintes passos:

- (a) Baixar a Ferramenta de Formatação da Associação SD acessando o link https://www.sdcard.org/downloads/formatter_4/eula_windows/;
- (b) Instalar e executar o software no notebook/computador utilizado;
- (c) Configurar a opção “*FORMAT SIZE ADJUSTMENT*” para *ON* no menu de *OPTION*;
- (d) Verificar se o cartão inserido é o mesmo selecionado pela ferramenta;
- (e) Selecionar a opção *FORMAT*.

Tendo configurado o cartão de memória no formato correto, basta apenas descompactar e copiar o arquivo NOOBS, baixado anteriormente, e grava-lo na raiz do cartão. Assim, o dispositivo de memória é inserido no RPI3.

Realizados os passos anteriores, energiza-se o RPI3 por meio da fonte de alimentação. Neste caso, foi utilizado uma fonte própria do minicomputador, mas poderia ser usado também uma fonte comum de celular, uma vez que ele possui entrada micro USB universal. É importante ressaltar que a fonte forneça alimentação de 5V e 700mA no mínimo, caso contrário o desempenho do RPI3 pode ser afetado. Quando o RPI3 é energizado, com o sistema operacional ainda não instalado, é possível se deparar com uma tela a qual proporciona a escolha do sistema operacional desejado, além do idioma do software.

Tendo selecionado o sistema operacional a ser instalado, basta aguardar enquanto o processo é feito. Neste passo, é interessante que RPI3 esteja conectado a um notebook/computador contendo acesso à internet por meio de um cabo *RJ45*, para que, durante o processo de instalação as atualizações de softwares sejam feitas automaticamente. Caso contrário elas podem ser feitas após este passo de forma individual. Ao término da instalação, o RPI3 será reiniciado automaticamente e se ligará como um computador comum.

Ao iniciar o RPI3, configura-se data, hora, idioma, nome de usuário, entre outras configurações básicas de um computador. Além disso, nota-se que ele

vai estar repleto com softwares básicos apropriados, um deles e o mais importante para este trabalho é *Python*, onde pode ser desenvolvido diversos programas e configuradas várias funções. Para efetuar o desligamento do minicomputador, basta selecionar a opção *shutdown* no menu. Em caso de desligamento direto pela fonte, a plataforma pode ter seu funcionamento comprometido.

3.1.2. Configurações das Portas GPIO

O software de programação que permite acessar e manipular as portas GPIO contidas no RPI3 é o *Python*. Antes de simplesmente programar as funções de cada porta, é necessário efetuar a instalação da biblioteca que diminui a complexidade de controle dos pinos. Essa tarefa pode ser feita simplesmente adicionando a seguinte linha de código ao programa:

```
apt-get install python-rpi.gpio
```

Desta forma, o código Python utiliza do comando abaixo para poder importar esta biblioteca no programa:

```
import RPi.GPIO as GPIO
```

Assim, define-se a numeração dos pinos GPIO da placa no modelo padrão (BOARD). A configuração de cada pino é mostrada pela Figura 3.2. Esta função é programada por meio da seguinte linha de código:

```
GPIO.setmode(GPIO.BOARD)
```

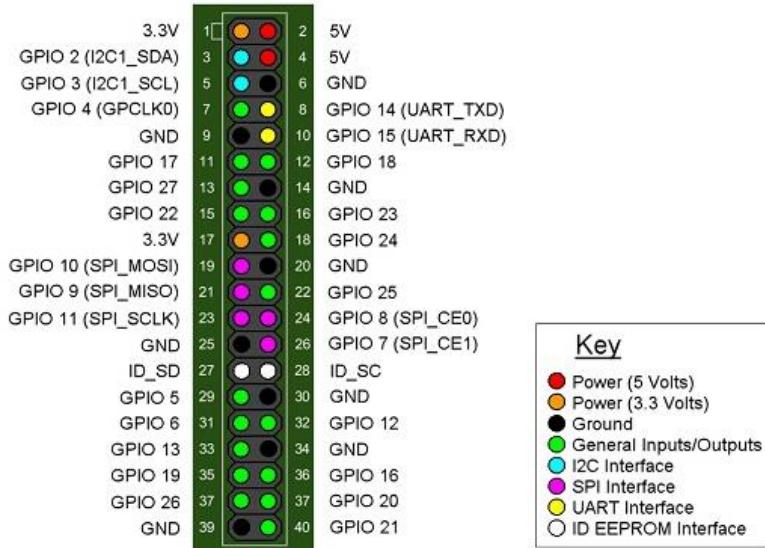


Figura 3.2: CONFIGURAÇÃO DOS PINOS GPIO.

3.1.3. Configuração da Interface SPI

O RPI3 possui um barramento serial chamado SPI (*Serial Peripheral Interface* – Interface serial periférica), o qual pode ser ativado nas configurações, uma vez que, por padrão de fábrica ele se apresenta desabilitado. Essa interface SPI é ativada nas GPIO's 19, 21, 23, 24 e 26, e é utilizada para que o Raspberry possa se comunicar não só com placas de expansão por meio de suas bibliotecas, mas também com outros dispositivos com barramento serial.

Neste projeto, a utilização da interface SPI se torna necessário devido à utilização do conversor A/D MCP3008, um circuito integrado SPI. Ele será o responsável pela conversão dos dados adquiridos no circuito eletrônico (será especificado nos próximos tópicos). A escolha deste modelo foi devido a sua grande compatibilidade com o RPI, onde um sinal analógico pode ser ligado a um dos canais (CH0 – CH7) do MCP3008 e este, ligado adequadamente ao RPI como mostrado pela Figura 3.3.

O MCP3008 é alimentado com uma tensão de 3.3 V, pois de acordo com o seu *datasheet*, para esta tensão de alimentação o ele trabalha com um *sample rate* de aproximadamente 100KHz, um valor de amostragem mais que satisfatório, pois é, em torno, 20 vezes maior que o valor de frequência mais alto a ser trabalhado. Deste modo, atende-se o critério de Nyquist para amostragem

do sinal sem risco de *aliasing*. Além disso, o MCP3008 possui 10 bits, isso faz com que ele tenha precisão similar de um Arduino UNO.

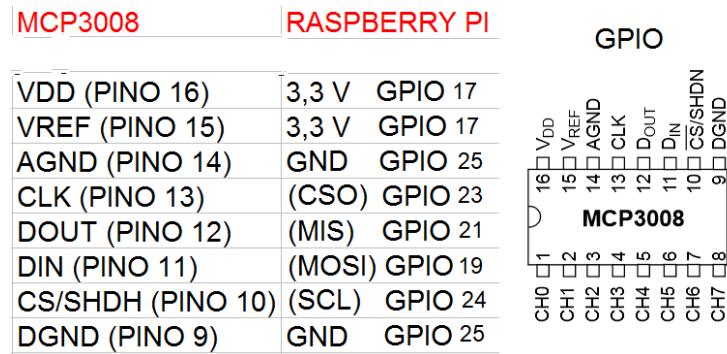


Figura 3.3: LIGAÇÃO MCP3008 AO RASPBERRY PI3.

A interface SPI do RPI pode ser habilitada de três maneiras diferentes. Neste projeto é abordada apenas uma delas, a qual foi utilizada e tida como a mais prática das 3. O processo é descrito nos seguintes passos:

1 – Com o terminal do RPI iniciado, digite a linha de código:

```
sudo raspi-config
```

O menu aparecerá conforme mostrado pela Figura 3.4. A opção nº5 é selecionada, uma vez que permite realizar configurações de conexões periféricas.

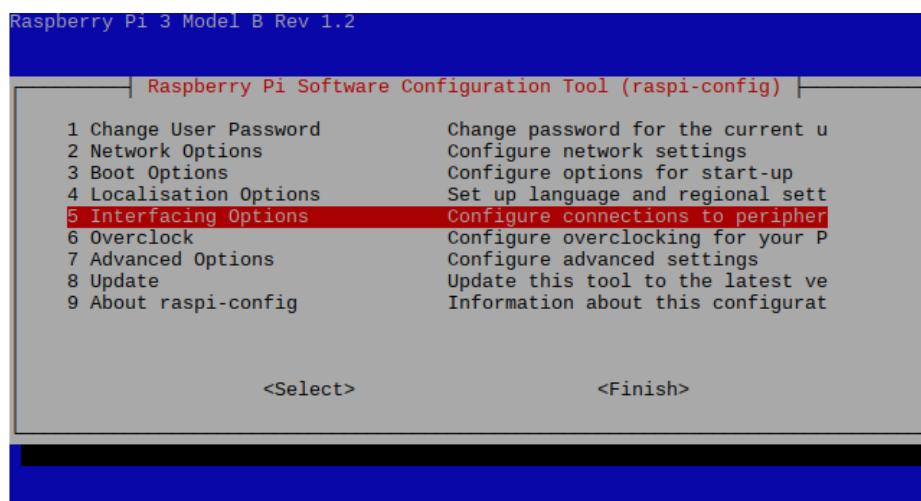


Figura 3.4: TELA DE CONFIGURAÇÃO DE SOFTWARE DO RASPBERR PI .

2 – Seguidamente, entra-se em um menu similar ao da Figura 3.5, onde a interface SPI é ativada ou desativada através da opção P4.

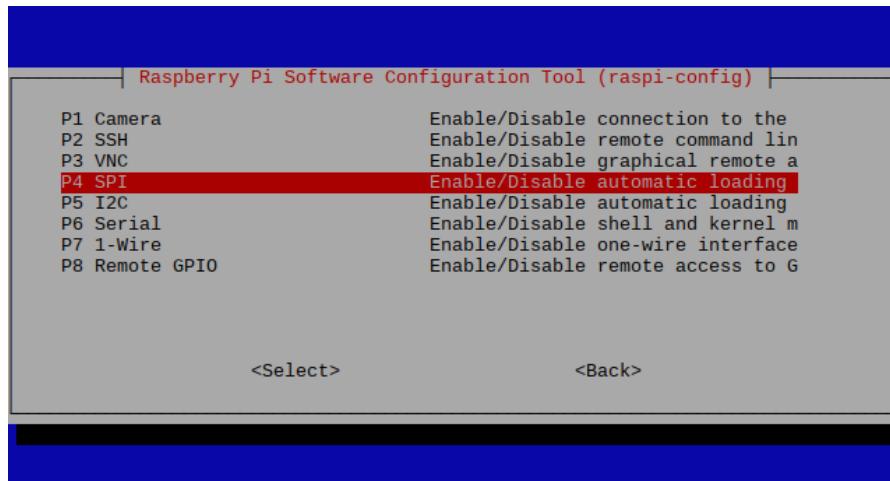


Figura 3.5: TELA DE CONFIGURAÇÃO PARA ATIVAÇÃO DA INTERFACE SPI.

3 - Por último, será solicitada a reinicialização do RPI. Basta selecionar a opção **Yes** para que esta ação seja executada. Ver Figura 3.6.

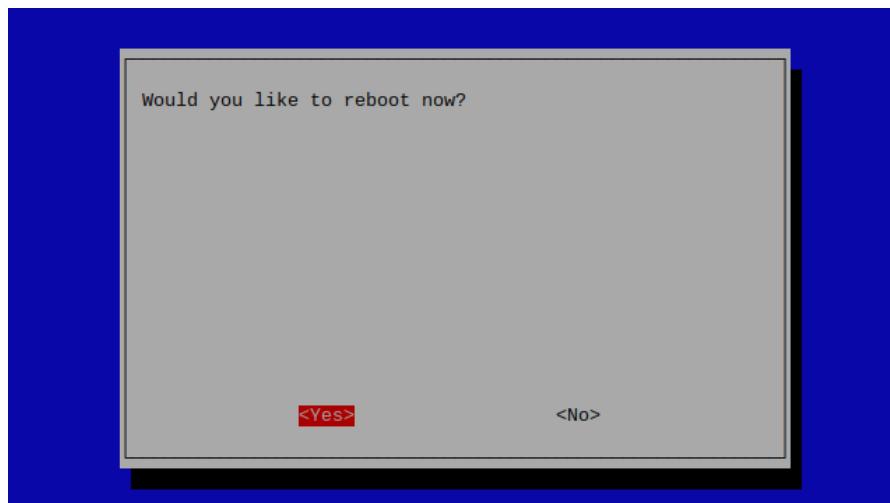


Figura 3.6: SELECIONE YES PARA REINICIALIZAR O RPI.

Tendo-se realizado estes passos, agora é necessário instalar a biblioteca para, por meio dela, configurar os pinos GPIO de forma correta. A biblioteca é nomeada como Adafruit Python MCP3008, pode ter seu código fonte encontrado no site *GitHub*. A instalação desse código fonte no RPI pode ser realizado através do terminal, utilizando algumas linhas de código, ver Figura 3.7.

Primeiro é preciso verificar se existem atualizações pendentes no sistema, para isso execute as duas primeiras linhas do código abaixo. Em seguida, o código fonte é buscado no site, como executado na linha de nº4. Agora basta executar a biblioteca e efetuar sua instalação, tais processos são mostrados nas linhas de nº5 e nº6 respectivamente.

```

1. sudo apt-get update
2. sudo apt-get install build-essential python-dev python-smbus
   git
3. cd ~
4. git clone
   https://github.com/adafruit/Adafruit_Python_MCP3008.git
5. cd Adafruit_Python_MCP3008
6. sudo python setup.py install

```

Figura 3.7: LINHAS DE CÓDIGO PARA INSTALAÇÃO DA BIBLIOTECA ADAFRUIT PYTHON MCP3008.

Ao final deste procedimento é mostrada, no terminal do RPI, a mensagem de instalação bem-sucedida, de forma similar à imagem mostrada abaixo, Figura 3.8.

```

pi@raspberrypi:~/Adafruit_Python_MCP3008
byte-compiling build/bdist.linux-armv6l/egg/Adafruit_MCP3008/_init_.py to _init_.pyc
byte-compiling build/bdist.linux-armv6l/egg/Adafruit_MCP3008/MCP3008.py to MCP3008.pyc
creating build/bdist.linux-armv6l/egg/EGG-INFO
copying Adafruit_MCP3008.egg-info/PKG-INFO -- build/bdist.linux-armv6l/egg/EGG-INFO
copying Adafruit_MCP3008.egg-info/SOURCES.txt -- build/bdist.linux-armv6l/egg/EGG-INFO
copying Adafruit_MCP3008.egg-info/dependency_links.txt -> build/bdist.linux-armv6l/egg/EGG-INFO
copying Adafruit_MCP3008.egg-info/requirements.txt -> build/bdist.linux-armv6l/egg/EGG-INFO
copying Adafruit_MCP3008.egg-info/top_level.txt -> build/bdist.linux-armv6l/egg/EGG-INFO
zip_safe flag not set; analyzing archive contents...
creating 'dist/Adafruit_MCP3008-1.0.0-py2.7.egg' and adding 'build/bdist.linux-armv6l/egg' to it
removing 'build/bdist.linux-armv6l/egg' (and everything under it)
Processing Adafruit_MCP3008-1.0.0-py2.7.egg
copying Adafruit_MCP3008-1.0.0-py2.7.egg to /usr/local/lib/python2.7/dist-packages
Adding Adafruit-MCP3008 1.0.0 to easy-install.pth

Installed /usr/local/lib/python2.7/dist-packages/Adafruit_MCP3008-1.0.0-py2.7.egg
Processing dependencies for Adafruit-MCP3008==1.0.0
Searching for Adafruit-GPIO==0.9.3
Best match: Adafruit-GPIO 0.9.3
Processing Adafruit_GPI0-0.9.3-py2.7.egg
Adafruit-GPIO 0.9.3 is already the active version in easy-install.pth

using /usr/local/lib/python2.7/dist-packages/Adafruit_GPI0-0.9.3-py2.7.egg
Searching for spidev==3.1
Best match: spidev 3.1
Processing spidev-3.1-py2.7-linux-armv6l.egg
spidev 3.1 is already the active version in easy-install.pth

Using /usr/local/lib/python2.7/dist-packages/spidev-3.1-py2.7-linux-armv6l.egg
Finished processing dependencies for Adafruit-MCP3008==1.0.0
pi@raspberrypi:~/Adafruit_Python_MCP3008 $ 
```

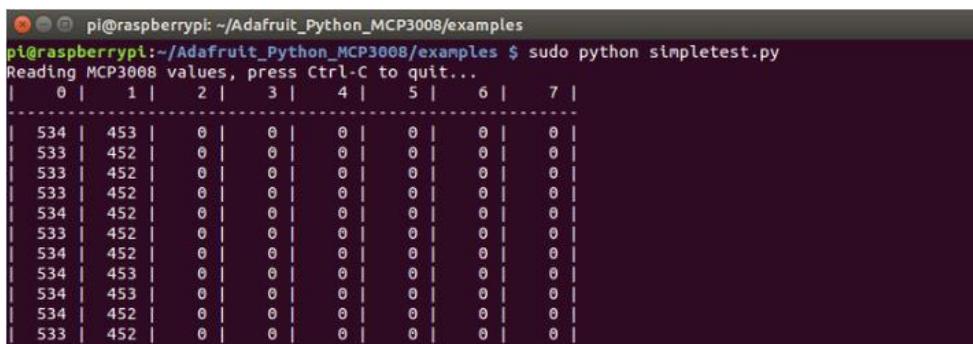
Figura 3.8: TELA APÓS A INSTALAÇÃO BEM-SUCEDIDA DA BIBLIOTECA ADAFRUIT PYTHON MCP3008.

Após a instalação, segue-se alguns procedimentos para operar a leitura do conversor. A biblioteca Adafruit Python MCP3008 disponibiliza um código base, chamado *simpletest.py*, o qual é capaz de fazer a leitura do conversor MCP3008.

Para utilizar este código, é necessário executar o comando subsequente no terminal do RPI.

```
cd ~ / Adafruit_Python_MCP3008 / exemplos  
sudo python differential.py
```

Após a executa destas linhas de código o terminal imprimirá em sua tela os valores lidos pelo conversor A/D. Como o MCP3008 possui 10 bits esses valores variam de 0 a 1023 bits. A Figura 3.9 ilustra essa ação.



A terminal window titled "pi@raspberrypi: ~/Adafruit_Python_MCP3008/examples". The command "sudo python simpletest.py" is run, followed by the message "Reading MCP3008 values, press Ctrl-C to quit...". Below this, a table displays 16 rows of 8-bit binary values. The columns are labeled 0 through 7. The values are mostly 0's, with some 1's appearing in the lower bits of the first few rows.

0	1	2	3	4	5	6	7
534	453	0	0	0	0	0	0
533	452	0	0	0	0	0	0
533	452	0	0	0	0	0	0
533	452	0	0	0	0	0	0
534	452	0	0	0	0	0	0
533	452	0	0	0	0	0	0
533	452	0	0	0	0	0	0
534	452	0	0	0	0	0	0
534	453	0	0	0	0	0	0
534	453	0	0	0	0	0	0
534	452	0	0	0	0	0	0
533	452	0	0	0	0	0	0
533	452	0	0	0	0	0	0
533	452	0	0	0	0	0	0

Figura 3.9: EXEMPLO DE LEITURA DO CONVERSOR MCP3008.

3.2. Determinação da Faixa de Frequência

A escolha da faixa de frequência a ser trabalhada é algo muito relevante para o desenvolvimento deste trabalho, pois é o parâmetro fundamental para elaboração do circuito eletrônico.

Assim, decidiu-se trabalhar com notas com uma frequência entre 30Hz e 4000Hz aproximadamente, o que equivale a 7 oitavas, variando no C1 (dó da primeira oitava) até C8 (Dó da última oitava). Essa faixa é escolhida pois, abrange notas da grande maioria dos instrumentos existentes na música ocidental, além disso, é em torno de onde o ouvido humano possuir maior sensibilidade ao som.

Para encontrar a frequência de cada nota musical, usa-se a relação matemática de $2^{(n/12)}$, onde n é a quantidade de ST's aumentados ou diminuídos a partir de uma frequência. Deste modo, multiplica-se essa relação pela nota base para aumentar um ST e divide-se caso seja desejado diminuir.

Como já é de conhecimento a nota A4 (lá da quarta oitava), possui uma frequência de 440Hz. Assim, é possível encontrar as outras notas da quarta oitava a partir de A4. A imagem 3.10, mostra de a relação de A4 com as outras notas da mesma oitava.

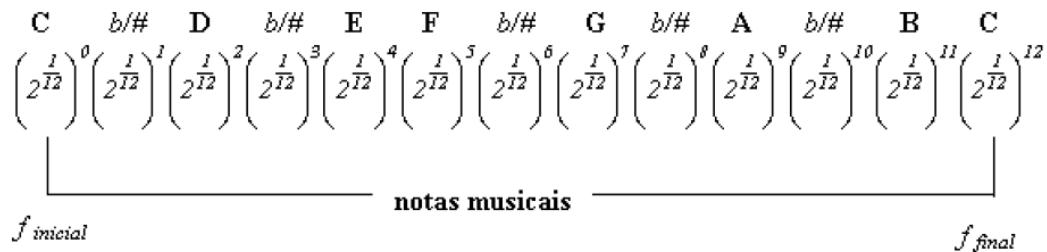


Figura 3.10: RELAÇÃO DAS NOTAS MUSICAIS EM UMA OITAVA.

Portanto, nota-se pela Figura 3.10 que, para calcular a frequência de C4 é preciso apenas dividir os 440Hz pela relação existente nas notas considerando $n = 9$. Esse processo se repete para o cálculo de todas as notas da oitava, utilizando suas respectivas relações.

3.3. Circuito de Aquisição de Dados

O circuito de aquisição de dados foi elaborado em basicamente 3 etapas. A primeira etapa consiste na captação do som por meio de um microfone de eletreto; na segunda etapa, é responsável por fornecer à onda de entrada um ganho pré-determinado e também garantir que a mesma opere dentro da faixa de operação deseja. Por último, utiliza-se um filtro passivo de terceira ordem do tipo passa-baixa para realizar um tratamento no sinal, eliminando frequências indesejadas e ruídos.

O primeiro passo do projeto do circuito se encontra na polarização e desacoplamento do microfone de eletreto. Para isto foi utilizado um resistor de 39 K Ω e um capacitor de 0.01 μ F.

O microfone foi alimentado por meio de uma fonte independente simples, com seus terminais em 5 V e GND. Afim de facilitar e tornar o circuito abrangente, o microfone foi ligado eletronicamente por meio de uma entrada JACK P2. Assim,

a qualquer momento o microfone pode ser alterado, caso haja algum mal funcionamento no mesmo.

O Resistor é ligado em série entre a alimentação e o positivo Jack P2, para limitar a corrente, que é o fator determinante para a capacidade de captação do microfone, isto é, resistores muito pequenos resultam em uma capacidade de captação relativamente alta. O capacitor por sua vez possui uma extremidade ligado ao positivo do conector e a outra é definida como saída do sistema. Ele é responsável pelo desacoplamento do microfone, estabelecendo um valor de tensão aproximadamente constante para que o sinal oscile. A Figura 3.11 (a) mostra a montagem eletrônica realizada e (b) mostra o esquemático da ligação equivalente a este circuito.

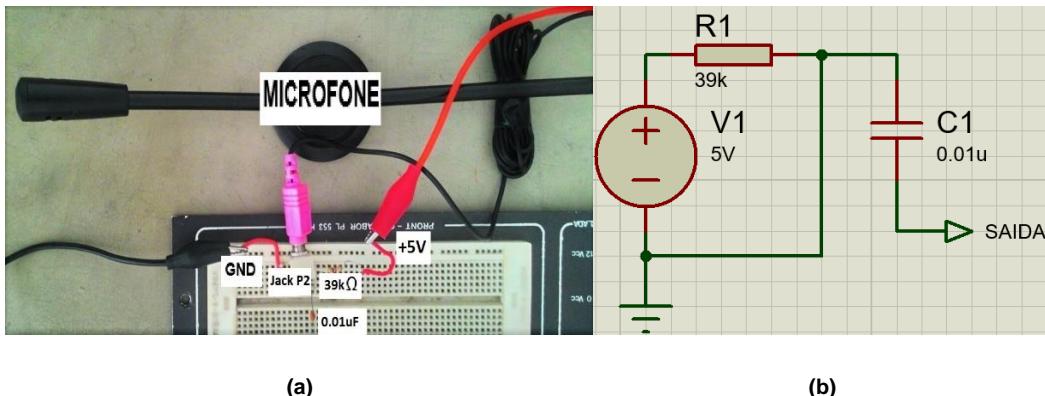


Figura 3.11: (a) MONTAGEM ELÉTRICA DA POLARIZAÇÃO DO MICROFONE. (b) ESQUEMÁTICO DE POLARIZAÇÃO DO MICROFONE.

A partir da forma de onda obtida a partir do desacoplamento do microfone, a qual será mostrado no capítulo posterior (Capítulo 4), foi possível elaborar um circuito responsável por gerar um ganho e um off-set. Esse processo foi necessário, pois o RPI3 pode receber apenas tensões de no máximo 3.3 V e o CI MCP3008 não suporta tensões acima de 5 V, então é desejável que o sinal esteja dentro dessa faixa para que o equipamento não seja sobrecarregado e comprometido. Além disso, o processo de digitalização não trabalha com sinais negativos, o que justifica o off-set realizado na onda. O projeto do circuito eletrônico foi elaborado utilizando o software Proteus 8 Professional. A Figura 3.12 mostra a montagem deste circuito, o qual será explicado detalhadamente nos próximos parágrafos.

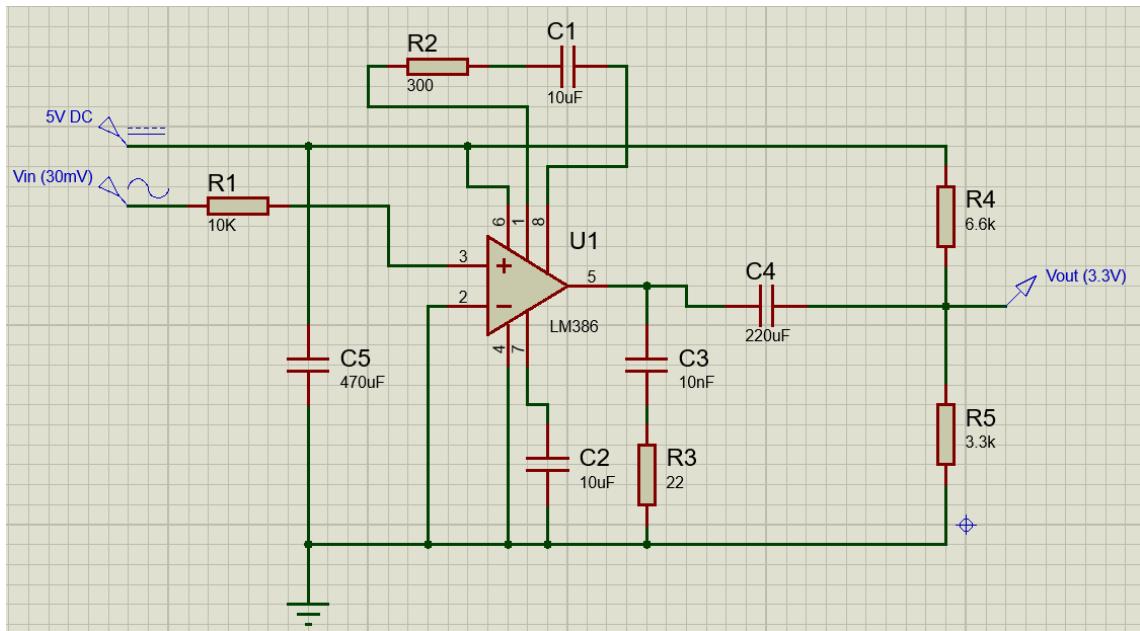


Figura 3.12: CIRCUITO GERADOR DE GANHO E OFF-SET NA ONDA.

Para desenvolvimento deste circuito, foi utilizado um amplificador operacional (AmpOp), modelo LM386. Este, diferente da maioria dos outros amplificadores, é um modelo próprio para se trabalhar pequenos sinais, além de ser um dos mais utilizados para tratar áudio de baixa potência. A alimentação deste sistema pode ser realizada, fisicamente, pelas portas GPIO de 5V do RPI, ou também por uma fonte independente, representado na Figura 3.12 pelo sinal contínuo 5VDC. A fonte senoidal $V_{in}(30mV)$ simula o sinal desacoplado do microfone, estabelecido com uma amplitude de 30mV. Este valor é uma aproximação feita a partir do teste de desacoplamento do microfone (Sessão 4.2.1).

Como pode ser visto na figura anterior, na parte inicial do circuito existe um resistor de 10 K Ω (R1) e um capacitor (eletrolítico) com capacidade igual a 470 μ F (C5). O primeiro foi colocado com a finalidade de causar uma pequena limitação na onda que chega do microfone, garantindo que ela não varie consideravelmente. Esse resistor pode ser substituído por um potenciômetro, o qual permitiria que o ajuste de sensibilidade da onda fosse feito manualmente. O segundo por sua vez, é colocado para o desacoplamento do circuito em geral e, desse modo, possibilitando realizar uma melhor análise CA, isto é, sem alguns ruídos proveniente da fonte de alimentação e do próprio circuito.

3.3.1. Amplificador Operacional LM386

Como já mencionado é um modelo próprio para se trabalhar sinais de baixa impedância. O LM386 possui três variantes: N-1; N-3; N-4. Para este projeto utilizou-se o N-1, uma vez que os critérios são atendidos por ele. Isto é, o LM386 N-1 pode ser alimentado com valores entre 4 – 12 V, além de possuir uma baixa corrente de repouso, visto que consome menos de 30 mW quando é alimentado com 5 V.

Assim, o segundo estágio do projeto eletrônico (Figura 3.12) é responsável por agir no ganho e também pelo off-set. Através da análise do *datasheet*, é possível ver que o amplificador LM386 tem seu trabalho interno dividido em 3 etapas, o *Input Stage* (Estágio de Entrada), *Voltage Amplifier Stage* (Estágio Amplificador de Tensão) e *Output Stage* (Estágio de Saída). A Figura 3.13 ilustra o circuito interno ao LM386 e suas etapas.

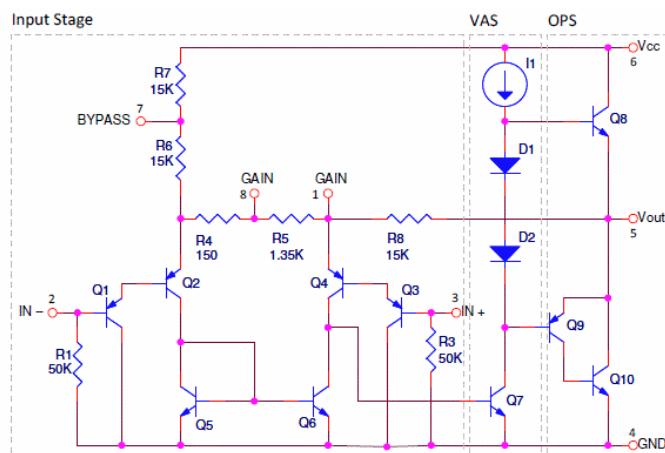


Figura 3.13: LM386 E SUAS ETAPAS INTERNAS PARA TRATAMENTO DO SINAL.

O *Input Stage* consiste em um amplificador transistorizado PNP coletor comum (Q1, Q3). O pino 2 é ligado ao *Ground* e o sinal de entrada, o qual sofrerá o ganho, é colocado junto ao pino 3. Assim, os resistores de 50 K (R1, R3), fazem com que a corrente de base de Q1 e Q3 seja drenada para o *Ground*. Isso acontece para que a impedância de entrada seja ajustada e não afete a polarização interna.

A segunda etapa (*Voltage Amplifier Stage*) é a mais importante, uma vez que ela é a responsável pelo ajuste de ganho. Sua análise é feita com o amplificador

estando no modo não operacional, portanto é possível calcular o ganho de tensão através do circuito resultante, mostrado pela Figura 3.14.

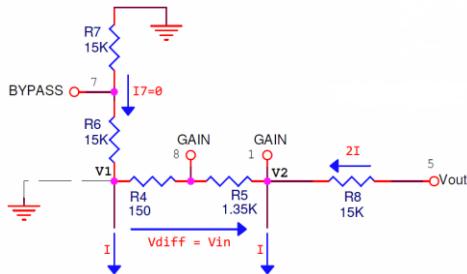


Figura 3.14: CIRCUITO DO MODO NÃO OPERACIONAL PARA CÁLCULO DE GANHO DE TENSÃO.

Neste caso, V_{diff} é o próprio valor de entrada V_{in} e como se trata de um alto valor de ganho, V_{out} é muito maior que V_{in} . Sendo assim, pode-se afirmar que:

$$\frac{V_{out} - V_{in}}{R_8} = 2I \quad (3.1)$$

$$V_{out} \gg V_{in} \quad (3.2)$$

$$\frac{V_{out}}{R_8} = 2I \quad (3.3)$$

Através da Figura 3.14 também é possível afirmar que:

$$I = \frac{V_{in}}{R_4 + R_5} \quad (3.4)$$

Logo;

$$Gv = \frac{V_{out}}{V_{in}} = 2 \left(\frac{R_8}{R_4 + R_5} \right) \quad (3.5)$$

Deste modo, é possível calcular o valor de ganho (G_v) do circuito por meio da eq. 3.5, para um caso em que não existe impedância entre os pinos 1 e 8. Logo, se não houver nada entre esses pinos, $G_v = 20$. Colocando um capacitor de 10 μF entre 1 e 8, o resistor que equivale a 1.35 K (R5) fica em curto e o ganho passa a ser igual a 200 V/V. Então, para ajustar o valor do ganho conforme o

desejado, é colocado um resistor em série com o capacitor entre os pinos 1 e 8 (ver Figura 3.12). Portanto, o ganho é calculado por meio da eq. 3.6.

$$Gv = 2 \left(\frac{Z_{1-5}}{150 + Z_{1-8}} \right) \quad (3.6)$$

Para este projeto, é considerado um sinal de entrada com oscilação média de 60mV de pico a pico. Como as entradas GPIO não suportam uma tensão maior que 3.3V e não é desejável sinais de valor negativo, devido a digitalização do mesmo, pode-se dizer q o ganho esperado é:

$$Gv = \frac{3.3}{0.06} \approx 55V/V \quad (3.7)$$

Com isso, é possível calcular o valor da resistência que se encontra em série com o capacitor de 10µF, ambos colocados entre os pinos de ajuste de ganho.

$$R_{Z_{1-8}} = \frac{2}{Gv} * Z_{1-5} - 150 \approx 350\Omega \quad (3.8)$$

Após o sinal de entrada ser submetido à etapa de ganho, ele passa pelo o último processo do circuito de ganho e *off-set*, onde é colocado um resistor de 10Ω em série com um capacitor de 10nF ligando o pino 5, pino de saída ao *Ground*. Isso é feito para evitar oscilações de alta frequência. Também é colocado um capacitor eletrolítico de 250µF entre a saída e o *off-set*, para que assim o componente CC possa ser filtrado. O *off-set* é dado pela razão das resistências de 6.6KΩ e 3.3KΩ, baixando a referência do sinal de 2.5V para 1.7V aproximadamente.

3.3.2. Filtro Ativo de 3^a Ordem

Seguidamente no desenvolvimento do circuito eletrônico, é elaborado um filtro passa-baixa, o qual será responsável por realizar uma limpeza no sinal e atenuar ruídos indesejados. Para projeto do filtro, foi utilizado o *software* FILTERPRO, fornecido pela TEXAS INSTRUMENTS. Este programa permite

que sejam estabelecidos os parâmetros do filtro e, através deles, é resultado esquemático e os componentes comerciais que compõem o filtro. Nele também é possível efetuar a análise de bode e o atraso de filtragem do sinal, que para este caso é importante avaliar.

Foi-se então projetado um filtro de 3^a ordem, uma vez que, como se trata de um circuito analógico não é viável uma ordem muito elevada, apesar de apresentar uma melhor eficiência, pois a atenuação do sinal na frequência de corte seria maior.

Como a maior frequência a ser identificada é 4186.009 Hz, que equivale ao C8, a frequência de corte foi estimada em torno de 4200 Hz. Mesmo que os sinais próximos a essa frequência sofram uma pequena atenuação, ainda se têm uma oscilação, pico a pico, considerável para efetuar a análise e reconstrução do sinal. Além disso, é considerado que o filtro ativo influencia no ganho do sinal. Sendo assim, buscou-se manter este ganho em torno de 1V/V aproximadamente.

Para execução de projeto de um filtro ativo real leva-se em consideração dois valores de frequência para corte do sinal, isto é, uma frequência determina a faixa de passagem (f_p) e a outra a faixa de bloqueio (f_s), o que é chamado de banda passante. A Figura 3.15 ilustra esse acontecimento.

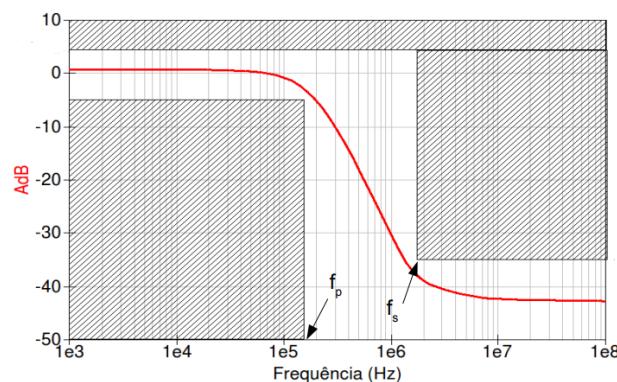


Figura 3.15: FAIXA DE PASSAGEM E FAIXA DE BLOQUEIO, FORMANDO A BANDA PASSANTE.

O filtro é projetado conforme a modelo de filtro de Butterworth, uma vez que esse modelo busca manter as frequências até a faixa de passagem com ganho aproximadamente igual, diferente de outros modelos, por exemplo Chebyshev,

que possui uma banda passante mais estreita, porém varia muito os valores de ganhos das frequências (BOYLESTAD, 2004). A Figura 3.16 mostra a comparação de resposta entre esses dois modelos de filtro ativo.

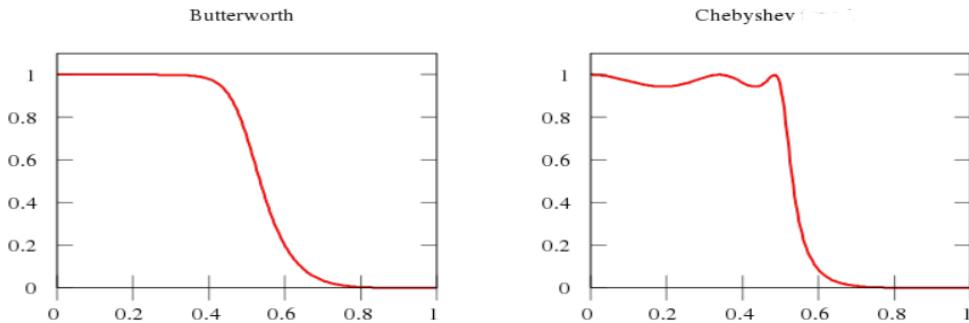


Figura 3.16: COMPARAÇÃO DE RESPOTA DOS MODELOS BUTTERWORTH E CHEBYSHEV.

A partir destas informações, é estabelecido os parâmetros: $f_p = 4000\text{Hz}$, $f_b = 4500\text{Hz}$ e $A \approx 1 \text{ V/V}$, os quais são passados ao *software* da TEXAS INSTRUMENTS. Assim, tem-se o resultado de projeto do filtro, como mostrado no circuito da Figura 3.17.

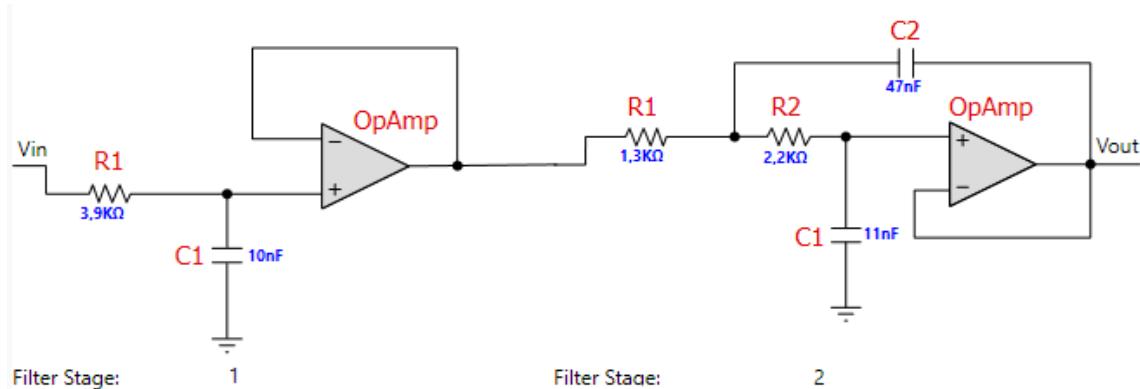


Figura 3.17: FILTRO ATIVO PASSA BAIXA DE BUTTERWORTH.

3.4. Código Desenvolvido para Aplicação de FFT e Identificação da Nota Musical.

O código do presente trabalho foi implementado utilizando a linguagem de programação Python 3, sendo assim, um fluxograma foi elaborado, afim de facilitar o entendimento de funcionamento do código. Ele é mostrado na imagem

a seguir pela Figura 3.18. O código na íntegra pode ser vislumbrado por completo no APÊNDICE C.

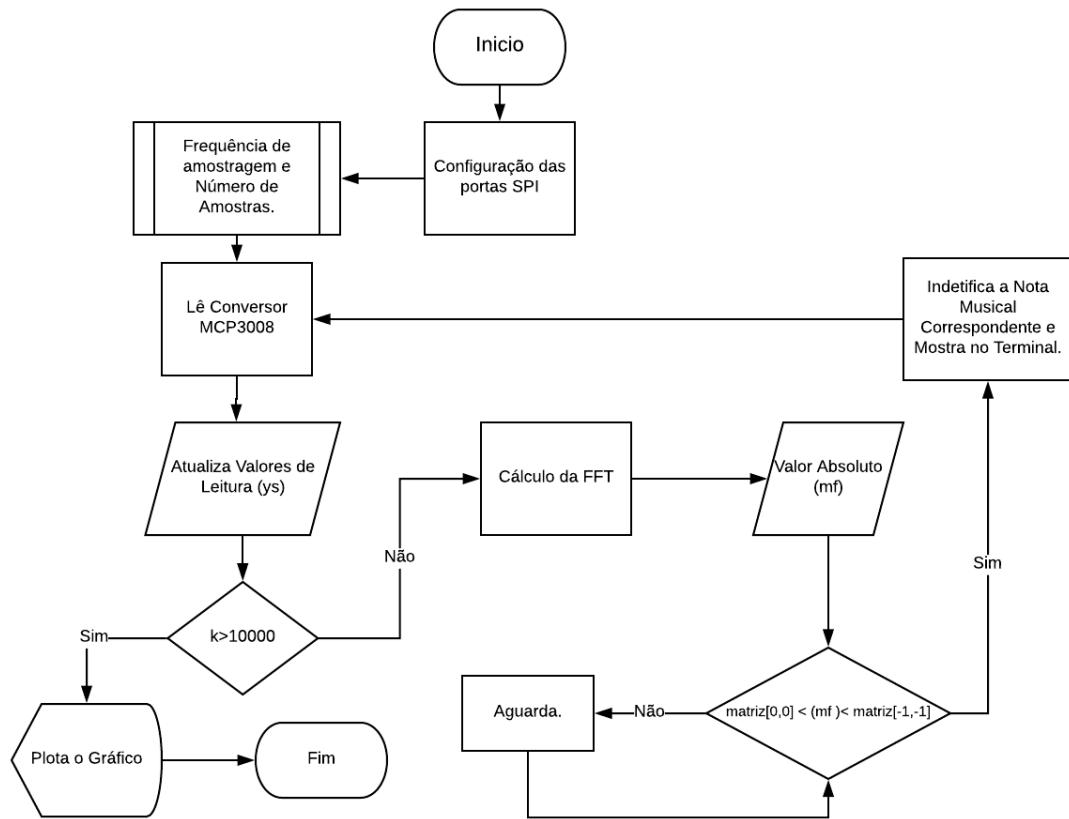


Figura 3.18: FLUXOGRAMA REPRESENTANDO A LÓGICA DO CÓDIGO DESENVOLVIDO.

Como é notado, inicialmente o código efetua as configurações necessárias para utilização SPI de *hardware* das portas GPIO. Após esse passo, são definidos os valores da frequência de amostragem (F_s) e do número de amostras (P_s), ao qual o processo será submetido.

Uma vez que esses ajustes são realizados, efetua-se leitura do conversor A/D MCP3008. Os valores são então armazenados em um vetor ys com tamanho equivalente a P_s . Neste instante, o processo entra em *loop*, de modo que a condição de parada é estabelecida por K , ou seja, K é o número total de amostras a serem lidas. Assim, quando este limite é atingido o processo de leitura é parado e o gráfico, contendo a última janela de leituras é plotado na tela. Caso o limite não seja atingido, o vetor contendo o sinal reconstruído é submetido ao cálculo

de sua *FFT*. O trecho do código contendo estas partes pode ser visto na Figura 8 do APÊNDICE C.

Dando continuidade a lógica de funcionamento do programa, a função responsável pelo cálculo da *FFT*, chamada de ‘*spectrum*’ (Figura 7, APÊNDICE C), retorna em sua saída o valor de frequência absoluto dentro desta janela de amostras, ou seja, a frequência a ser identificada como uma das notas musicais.

Esse valor é então comparado com a matriz, nomeado por ‘*mapa*’ (Figura 9, APÊNDICE C). Essa matriz é feita tendo por base o código mostrado na Figura 1 do APÊNDICE A, usado para mapear as notas de acordos com sua altura e o valor de suas respectivas frequências. Também foi acrescentada a está lógica função do Python *argmin()*, responsável por realizar a aproximação para o valor de frequência mais próximo existente na matriz. Isto é, caso o a frequência absoluta identificada não seja igual a nenhum dos valores contidos na matriz ‘*mapa*’, e esteja dentro da faixa de valores analisados, faz-se a aproximação para o valor mais próximo.

Após a frequência absoluta passar pelo reconhecimento, tem-se duas opções, como é ilustrado no fluxograma. Para uma situação em que a frequência não está dentro da faixa de valores analisados, entre 30 Hz a 4000 Hz aproximadamente, o programa não realiza nenhuma tarefa e aguarda até a próxima análise. Em caso contrário, a nome e altura da nota musical são mostrados na Terminal do RPI, além disso, simultaneamente, parte dos valores do vetor *ys* são atualizados de acordo com a leitura efetuada pelo MCP3008.

4. Resultados

Este capítulo está destinado a mostrar os resultados encontrados durante ao longo do desenvolvimento deste trabalho. Foi determinado obter como resultado primordial a identificação das notas musicais emitidas pelas cordas de um violão. Os resultados intermediários consistem na avaliação do sinal após sofrer o ganho, eficiência do filtro ativo e integridade da forma de onda digitalizada.

4.1. Mapeamento de Frequências

Com base nos passos, descritos na metodologia, para encontrar as frequências das notas de uma oitava, no caso, a quarta oitava, basta duplicar ou dividir pela metade cada para alternar as oitavas. Isso se deve, pois como já mencionado, a relação entre uma nota e a mesma uma oitava acima é de 1 para 2. Deste modo, cada nota é mapeada de acordo com sua frequência utilizando métodos computacionais, conforme o script mostrado pelo APÊNDICE A ao final deste documento. A Tabela 4.1 mostra as notas e suas respectivas frequências, calculada em cada oitava, para uma faixa de 7 oitavas.

NOTAS		1	2	3	4	5	6	7
1	C	32.703196	65.406391	130.81278	261.62557	523.25113	1046.5023	2093.004
2	#/b	34.647829	69.295658	138.59123	277.18263	554.36526	1108.7305	2217.461
3	D	36.708096	73.416192	146.83238	293.66477	587.32954	1174.6591	2349.3181
4	#/b	38.890873	77.781746	155.56349	311.12698	622.25397	1244.5079	2489.0159
5	E	41.203445	82.406889	164.81378	329.62756	659.25511	1318.5102	2637.0205
6	F	43.653529	87.307058	174.61412	349.22823	698.45646	1396.9129	2793.8259
7	#/b	46.249303	92.498606	184.99721	369.99442	739.98885	1479.9777	2959.9554
8	G	48.999429	97.998859	195.99772	391.99544	783.99087	1567.9817	3135.9635
9	#/b	51.913087	103.82617	207.65235	415.3047	830.6094	1661.2188	3322.4376
10	A	55	110	220	440	880	1760	3520
11	#/b	58.27047	116.54094	233.08188	466.16376	932.32752	1864.655	3729.3101
12	B	61.735413	123.47083	246.94165	493.8833	987.7666	1975.5332	3951.0664

Tabela 4.1: NOTAS MUSICAS E SUAS FREQUÊNCIAS EM UMA FAIXA DE 7 OITAVAS.

Por último, é encontrada a frequência de $C8 = 4186.009$ Hz, que não foi acrescentada a tabela por questões estéticas. A variação entre as notas no meio físico acontece de forma gradativa e não por meio de saltos, ou seja, se uma corda vibra emitindo o som relativo à C1 e, em seguida vibra em C#1, a variação da frequência ocorre gradualmente no tempo. Isso remete à necessidade de estabelecer uma margem de transição entre as notas consecutivas. Essa margem é estabelecida como sendo a metade da diferença entre as duas notas consecutivas.

4.2. Captação de Áudio

Para realização dos testes iniciais de funcionamento do microfone, foi utilizado um osciloscópio do Laboratório de Eletrônica do CEFET-MG. Efetuou-se a montagem do circuito de polarização e desacoplamento do microfone. Este foi então alimentado e teve sua saída ligada a uma das ponteiras do equipamento de medição e a outra ponteira foi ligada ao GND. Assim, foi emitido a partir de um celular, o áudio de uma senóide equivalente à 1 KHz. O resultado encontrado na medição é mostrado pela Figura 4.1.

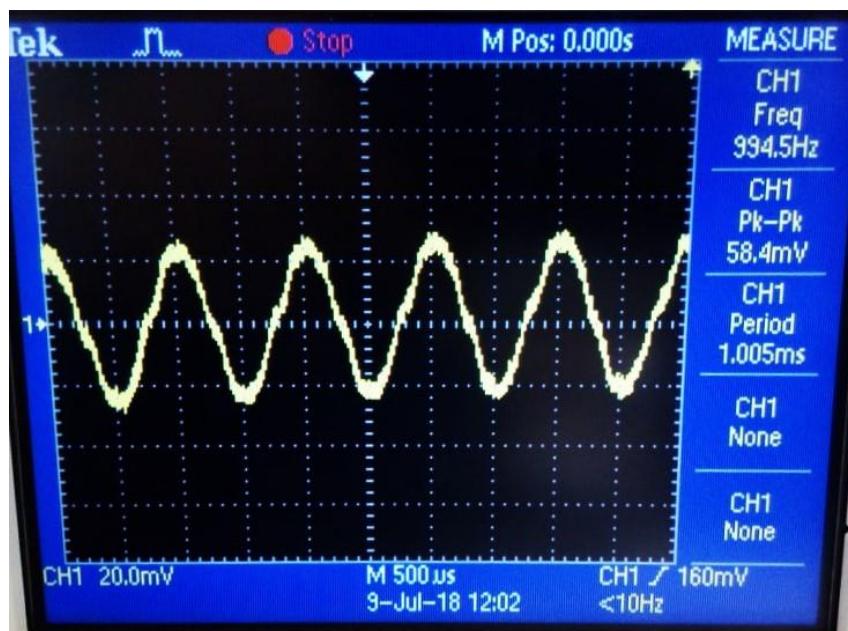


Figura 4.1: FORMA DE ONDA DO DESACOPLAMENTO E POLARIZAÇÃO DO MICRFONE.

4.2.1. Circuito de ganho e off-set

Nota-se então que a tensão oscilou em torno de um referencial fixo com uma forma senoidal conforme o esperado. Além disso, percebe-se que a onda possui um pico a pico equivalente à 60 mV, valor que é utilizado como base para projetar o circuito responsável pelo ganho e off-set do sinal de entrada, como já foi explicado no capítulo anterior deste documento. O ambiente de simulação, é então, montado no software Proteus 8 e logo em seguida o teste é executado (Figura 4.2).

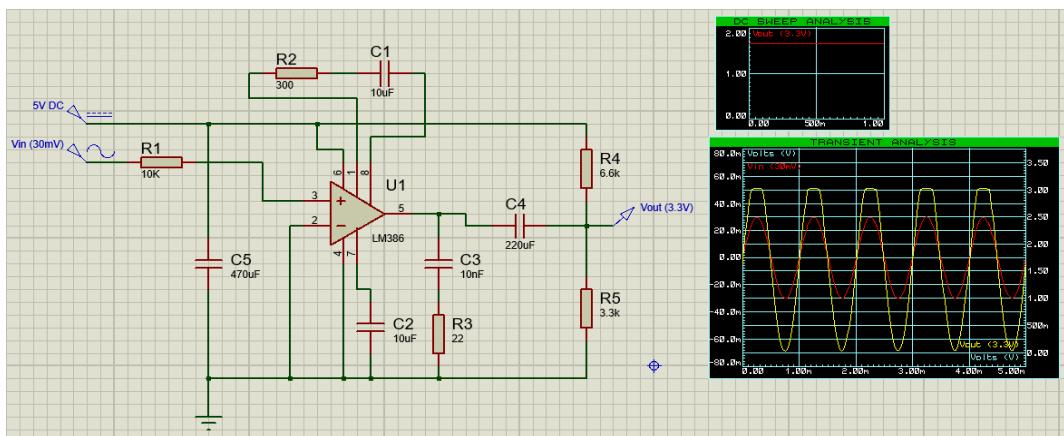


Figura 4.2: AMBIENTE DE SIMULAÇÃO UTILIZADO PARA TESTES NO PROTEUS 8.

Nestes testes é utilizado como entrada uma senóide de 30mV de amplitude, representado por $V_{in}(30mV)$ e a saída é coletada no ponto nomeado $V_{out}(3.3V)$. A interface chamada de “*DC SWEEP ANALYSIS*” é responsável por mostrar o resultado CC do circuito, para que assim seja possível analisar o valor de tensão onde está localizado o off-set. Por sua vez, a outra interface (“*TRANSIENT ANALYSIS*”) efetua a análise CA, permitindo relacionar o sinal de entrada e o sinal de saída.

Para uma análise mais detalhada de cada interface, as mesmas são mostradas pelas Figura 4.3 e Figura 4.4.



Figura 4.3: RESPOSTA CC OBTIDA NA SIMULAÇÃO.

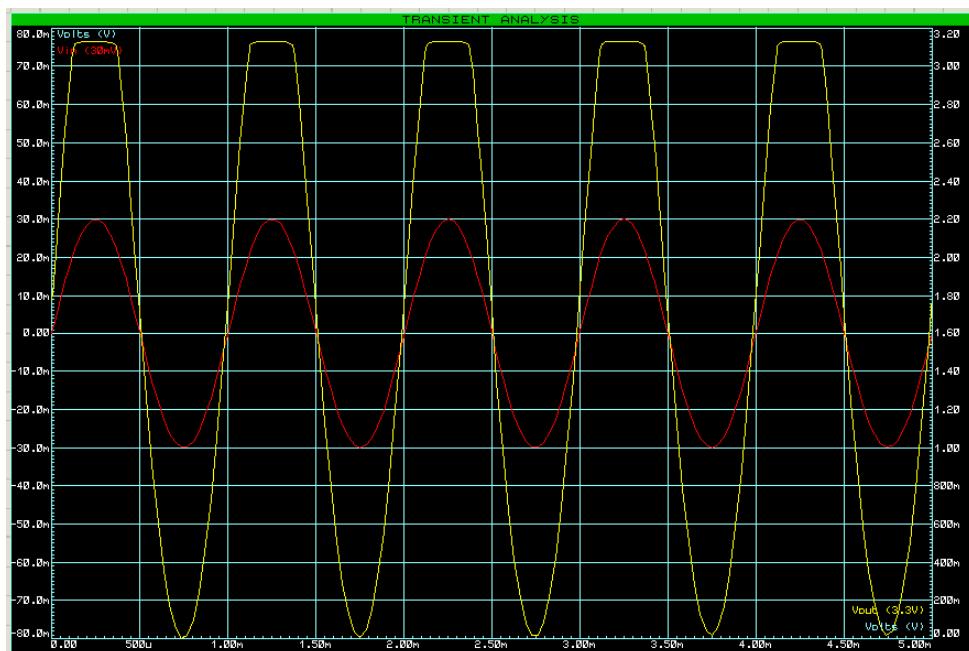


Figura 4.4: RESPOSTA CA OBTIDA NA SIMULAÇÃO.

Portanto é verificado que os resultados adquiridos foram próximos ao esperado. Através da Figura 4.3 é perceptível que o off-set se instaura em um valor de tensão um pouco menor do que 1.7 V. Já a partir da Figura 4.4, é importante ressaltar que são feitas 3 análises. A primeira é baseada no eixo horizontal inferior do gráfico. Observa-se que ambas as formas de onda completam um ciclo em 1.00 ms, isso significa que, a frequência de 1000Hz tida na entrada não sofreu deformação ou interferências negativas ao longo do processo.

As outras duas análises desta interface são feitas segundo os eixos verticais esquerdo e direito. Logo, o sinal de entrada (cor vermelha) é referenciado pelo eixo vertical da esquerda, mostrando os 60 mV pico a pico, e o sinal de saída (cor amarela) tem seus valores mensurados pelo eixo da direita. Assim, pode-se ver que a senóide amarela possui aproximadamente 3.15 V de pico a pico, constatando um ganho próximo de 52.5 V/V.

Posteriormente à simulação, é efetuada a montagem do circuito eletrônico em uma *protoboard* para realização de testes práticos. É importante ressaltar, que valores de alguns resistores podem sofrer uma suave alteração, devido a imprecisão de 5% desses componentes, afim de alcançar melhorias na resposta prática, fotos na montagem real do circuito podem ser vistas no APÊNDICE B.

Inicialmente o teste foi feito com uma onda senoidal de 1000 Hz, já que esta foi usada de base para simulação. A Figura 4.5 mostra o resultado captado pelo osciloscópio.

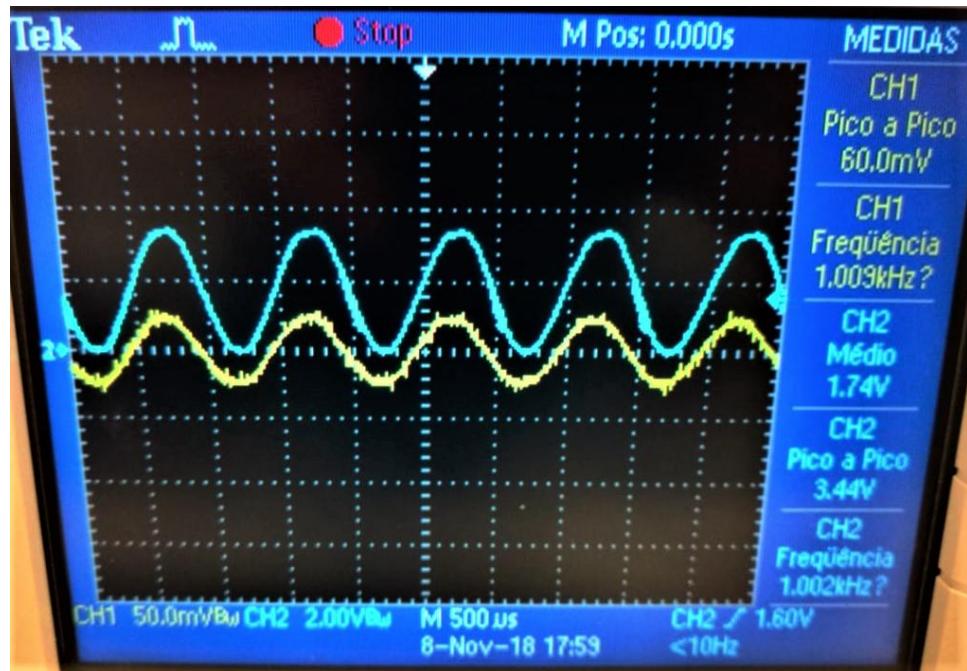


Figura 4.5: RESPOSTA DO CIRCUITO REAL QUANDO SUBMETIDO A UM SINAL DE 1KHZ.

Como é percebido, a senóide amarela representa o sinal de entrada e a de cor azul se refere ao sinal de saída do sistema (Figura 4.1.5). Ao lado direito da imagem, é mostrado na tela do osciloscópio os valores medidos dos sinais de acordo com suas respectivas cores. Uma vez que o resultado se apresentou adequado, o próximo passo consiste na montagem elétrica do filtro e seus eventuais testes.

4.2.2. Filtro Ativo de 3^a Ordem

A simulação deste elemento do projeto é feita pelo próprio *software* TEXAS INSTRUMENTS. A análise de eficiência do filtro pode ser feita pelo Diagrama de Bode do mesmo. Esse diagrama representa o ganho de fase e de módulo por meio de um gráfico Frequência (Hz) x Ganho (dB). Por meio da Figura 4.6 pode-se ver que para -3 dB de ganho, isto é, o decaimento de ganho do sinal na frequência de corte do filtro, apresenta-se um valor de frequência próximo a 4200Hz, como esperado.

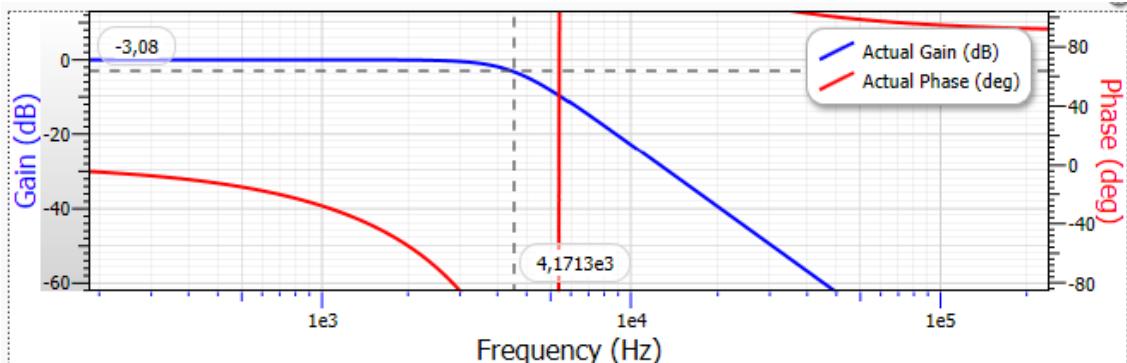


Figura 4.6: DIAGRAMA DE BODE DO FILTRO PROJETADO.

Através do *software* fornecido pela TEXAS INSTRUMENTS também é possível obter o gráfico de atraso do sinal ao passar pelo filtro. Para este trabalho especificamente, esse é um parâmetro importante a ser avaliado, visto que o tempo de atraso pode resultar diretamente na reconstrução do sinal. A Figura 4.7 mostra a relação desse atraso e as frequências trabalhadas.

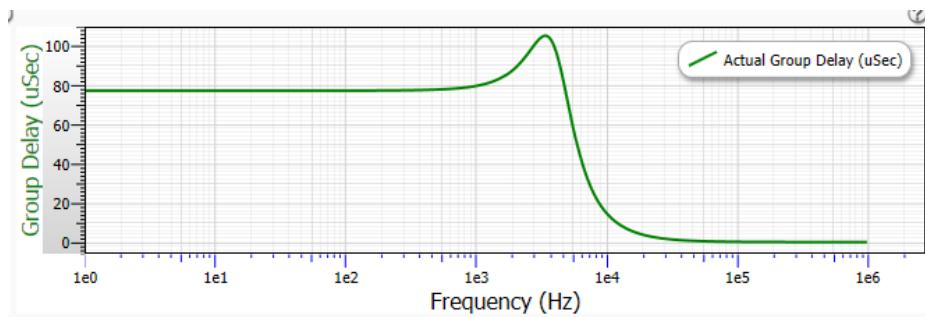


Figura 4.7: ATRASO DO SINAL QUANDO PASSA PELA FILTRAGEM.

Como é notado, o maior valor de atraso que o sinal pode vir a ter, devido a utilização deste filtro, é cerca de 100 μ s, o que equivale a 1 KHz. Como a taxa de amostragem tem que ser no mínimo 8 KHz, ou seja, duas vezes maior que a maior frequência amostrada, é válido afirmar que o atraso de filtragem do sinal não interfere na reconstrução do mesmo.

Para a validação prática do filtro foram utilizadas mais três formas de onda, além da de 1 KHz, são elas: 2 KHz, 4KHz e 8KHz. As Figura 4.8, Figura 4.9 e Figura 4.10 mostram os resultados encontrados para cada uma das frequências, respectivamente.

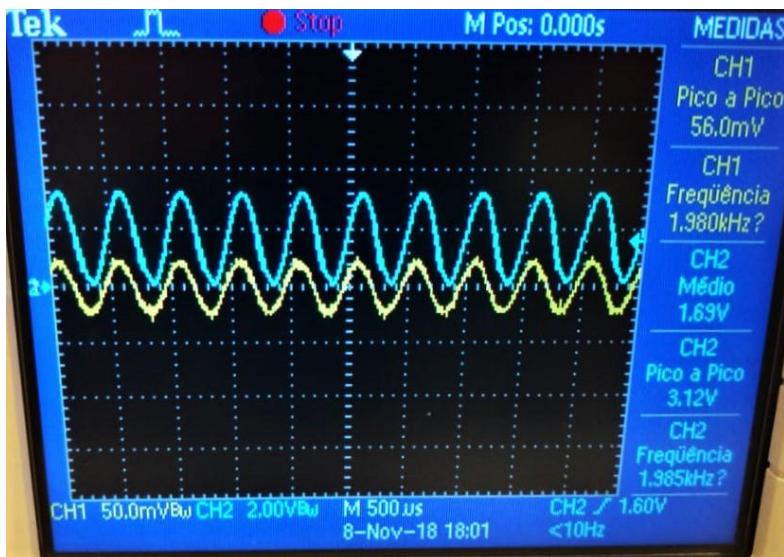


Figura 4.8: RESPOSTA DO CIRCUITO GANHO, OFF-SER E FILTRO PARA UMA ENTRADA DE 2KHZ.

É apercebido por meio da Figura 4.8 que o filtro manteve o ganho aproximadamente igual, além de não ter atenuado o sinal significativamente, quando é submetido a um sinal de 2 KHz, como previsto no projeto do filtro.

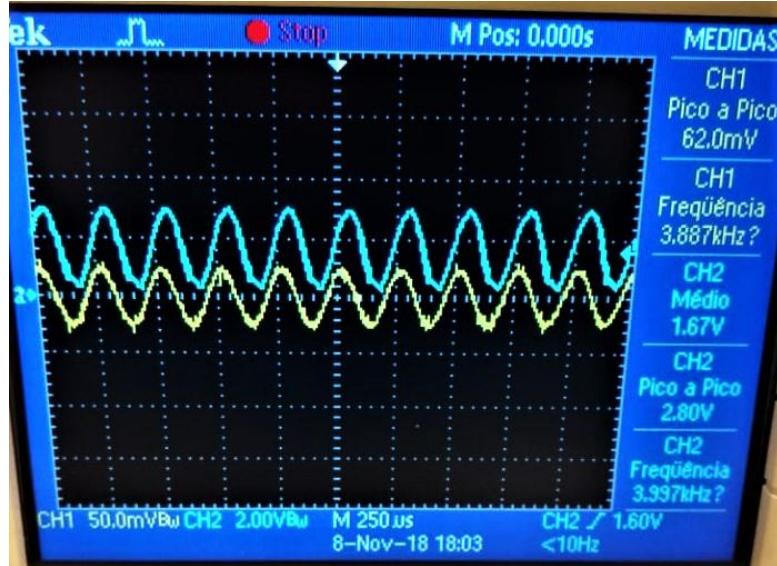


Figura 4.9: RESPOSTA DO CIRCUITO GANHO, OFF-SER E FILTRO PARA UMA ENTRADA DE 4KHZ.

Já quando o sistema é imposto a uma frequência de 4Khz, neste caso, a frequência de corte, pode ser analisado uma atenuação considerável no sinal, além de possuir defasamento, uma vez que o atraso. O motivo disto é o fato de o sinal sofre -3 dB na frequência de corte, como já foi mencionado. A relação do ganho em decibéis (dB) e em Volts (V), é dado pela eq. 4.1. Pelos dados da simulação tem-se que para 3981,072 Hz uma atenuação de -2.603 dB.

$$V_{atenuado} = 10^{\left(\frac{A_{dB}}{20}\right)} * V_{referencia} \quad (4.1)$$

Deste modo, calcula-se um sinal atenuado com aproximadamente 2.5 V pico a pico. Agora, quando o sistema é submetido ao sinal de 8000Hz, é possível assinalar uma atenuação muito maior (Figura 4.10), quando este é comparado com sinal inicial não atenuado. Desse modo, pode-se considerar que o filtro apresentou resultados aceitáveis para este trabalho.

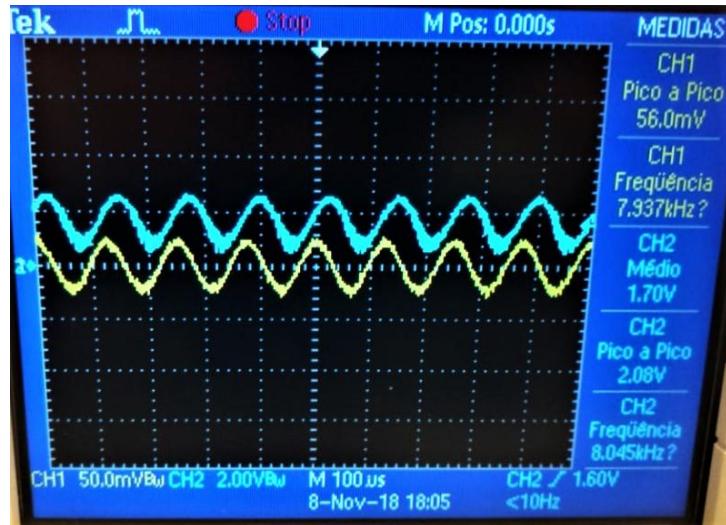


Figura 4.10: RESPOSTA DO CIRCUITO GANHO, OFF-SER E FILTRO PARA UMA ENTRADA DE 8KHZ.

4.3. Reconstrução do sinal

O último teste a ser realizado é a reconstrução do sinal. Tendo efetuado a ratificação do circuito eletrônico de aquisição de dados, pode-se então conectar a saída do filtro em um dos canais de entrada do conversor MCP3008, neste caso foi utilizado o canal zero (PINO 8).

Através do código desenvolvido, visto no APÊNDICE C, é possível selecionar a frequência de amostragem (F_s). Como o maior valor é a ser reconstruído é 4186 Hz, F_s é estabelecido em 10 KHz, afim de satisfazer os critérios de Nyquiste. Para realização dos testes, foram ajustados também as variáveis P_s e K . Como já mencionado, o primeiro é utilizado para selecionar um determinado número de amostras e, por meio dessas, plotar um gráfico contendo a forma de onda reconstruída. Já o segundo, faz com que a leitura do conversor seja cessada, encerrando o processo de aquisição de sinal, baseando no valor de P_s , e permitindo que a plotagem aconteça. A Figura 4.11 exibi um trecho do código contendo estes parâmetros.

```

25  Fs = 15000 #Frequencia de amostragem
26  Ps = 100 #Numero de pontos adquiridos
27  ys = [] # cria lista vazia
28  ts = [] #
29  fs = [] #
30
31  for _ in range(Ps): #Armazenamento de um ciclo do sinal no vetor
32      y = mcp.read_adc(0) #realiza leitura do conversor
33      ys.append(y) # armazena os valores no em ys
34
35  k = 0
36  while True: #Condicao de parada caso seja desejado
37      k+=1
38      if k > 10000:
39          break
40

```

Figura 4.11: PARÂMETROS PRÉ-ESTABELECIDOS PARA EXECUÇÃO DO CÓDIGO E EVENTUALMENTE DO TESTE.

Após estes processos, foi validado a reconstrução da onda sonora captada. Para isso, foram-se utilizados dois sinais senoidais, o primeiro com frequência igual a 500 Hz e o outro com frequência de 4000 Hz. Esses valores são escolhidos, pois tratam de situações com uma frequência mais baixa do que os sinais usados em testes anteriores (1 KHz) e também de uma frequência próximo à frequência de corte do filtro. Sendo assim, torna-se possível avaliar o sinal reconstruído em duas condições extremas. As Figura 4.12 e Figura 4.13 mostram os resultados adquiridos com base nas frequências de teste, 500 e 4000 Hz respectivamente, onde foi considerado um Ps de 100 amostras e Fs de 15 KHz para plotagem.

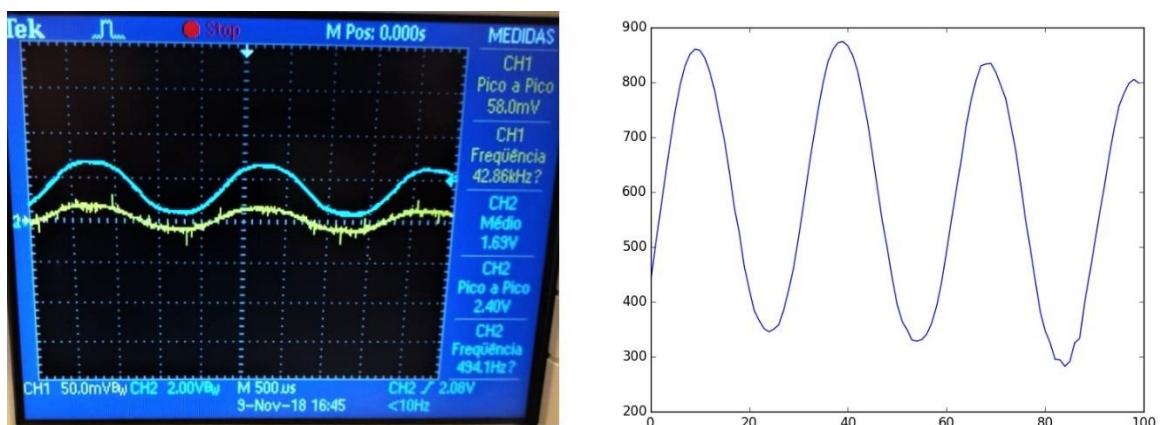


Figura 4.12: RECONSTRUÇÃO DA FREQUÊNCIA DE 500HZ.

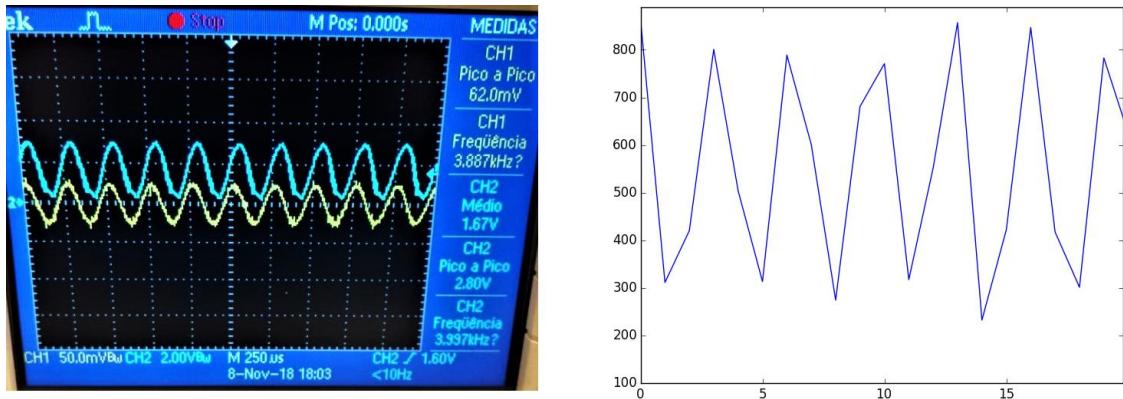


Figura 4.13: RECONSTRUÇÃO DA FREQUÊNCIA DE 4000Hz.

Em ambas as imagens é possível visualizar a medição da onda no osciloscópio, à esquerda, e o resultado obtido da reconstrução da mesma, à direita. Na medição da reconstrução do sinal, o eixo vertical mostra a variação em bits da onda reconstruída e o eixo horizontal o número de amostras obtidos. Pode-se encontrar a relação de número de amostras a cada ciclo do sinal, ou seja, a quantidade de pontos por período, e assim avaliar a confiabilidade da reconstrução. Essa relação é mostrada pela eq. (4.2):

$$T_{amostra/periodo} = \frac{\frac{1}{F_{signal}}}{\frac{1}{F_s}} \quad (4.2)$$

Logo, pode-se constatar que a frequência equivalente a 500 Hz, através da eq. 4.2, deve possuir aproximadamente 30 pontos amostrados, quando o sinal completa 1 período. Se for analisada a Figura 4.12 cuidadosamente, será possível visualizar que realmente existem torno de 30 amostras em um ciclo do sinal de 500Hz. O mesmo procedimento é feito para a onda de 4000 Hz. Neste caso, deve-se possuir cerca de 3.75, ou melhor, 4 amostras em um ciclo, como também pode ser notado na Figura 4.13. É possível perceber, junto ao último sinal amostrado, uma formação triangular da onda reconstruída. Isso implica no aumento da frequência de amostragem em testes futuros, para que os sinais de maior frequência tenham sua reconstrução menos deformada.

Seguidamente, a última etapa de experimentação deste trabalho, se conve na medição das notas musicais emitidos pelas cordas de um vilão comum. Para

este teste, a frequência de amostragem foi ajustada para 80 KHz e definiu-se um janelamento de 500 pontos ($Ps = 500$).

Neste último ensaio do trabalho, o nome e a altura das notas equivalentes a cada sinal são mostradas no terminal do RPI3, tendo como base os valores encontrados na Tabela 4.1. O programa desenvolvido (APÊNDICE C) foi executado, e foram tocadas as cordas soltas do violão, em um intervalo de aproximadamente 10 segundos. Esse tempo foi suficiente para que a leitura do sinal se estabilizasse, e o programa fosse interrompido, mostrando as notas lidas e o *plot* com o último janelamento do sinal.

As notas de um violão na afinação comum são: E2 (82.406Hz), A2 (110Hz), D3 (146.832Hz), G3 (195.997Hz), B3 (246.941Hz) e E4 (329,627Hz). Os resultados encontrados são mostrados pela Figura 4.14, Figura 4.15, Figura 4.16, Figura 4.17, Figura 4.18, Figura 4.19, da nota mais grave para nota mais aguda.

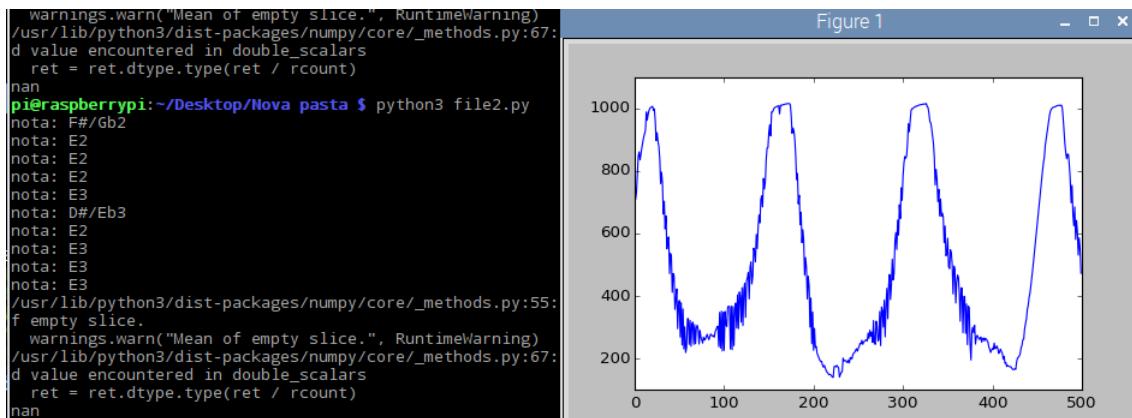


Figura 4.14: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA E2.

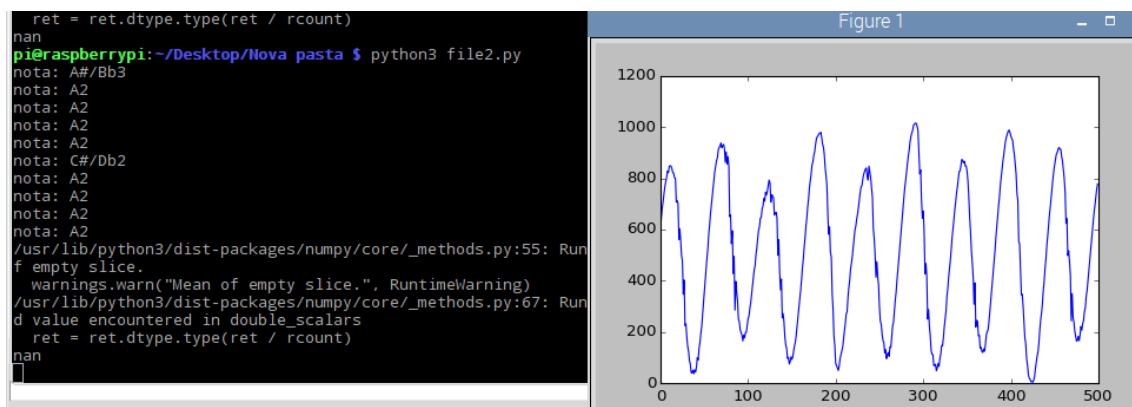


Figura 4.15: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA A2.

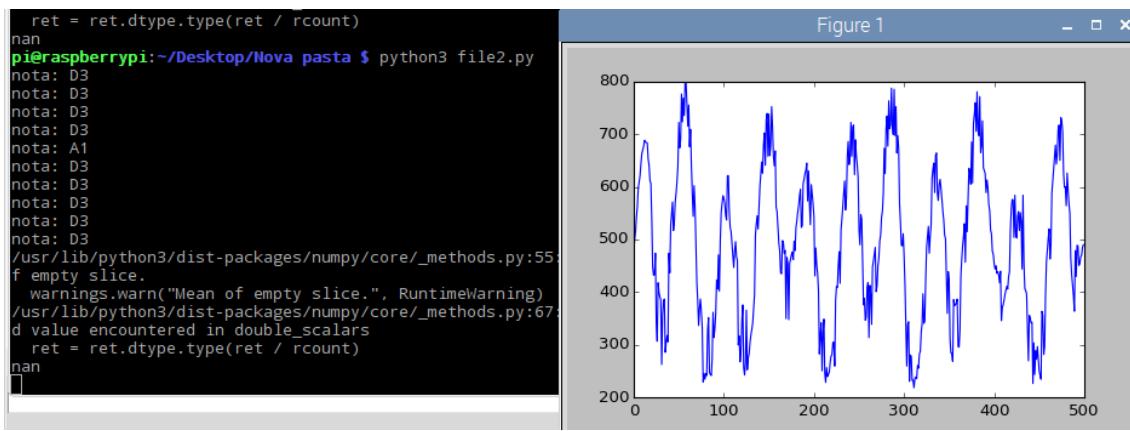


Figura 4.16: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA D3.

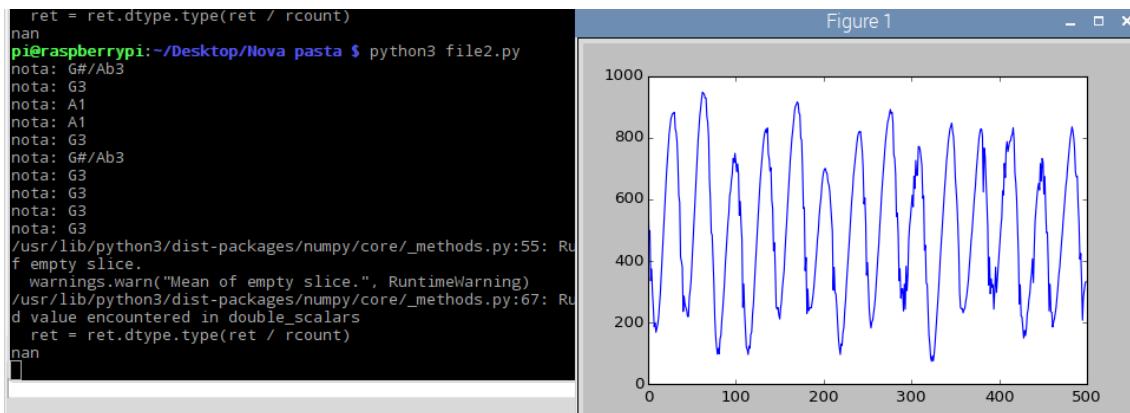


Figura 4.17: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA G3.

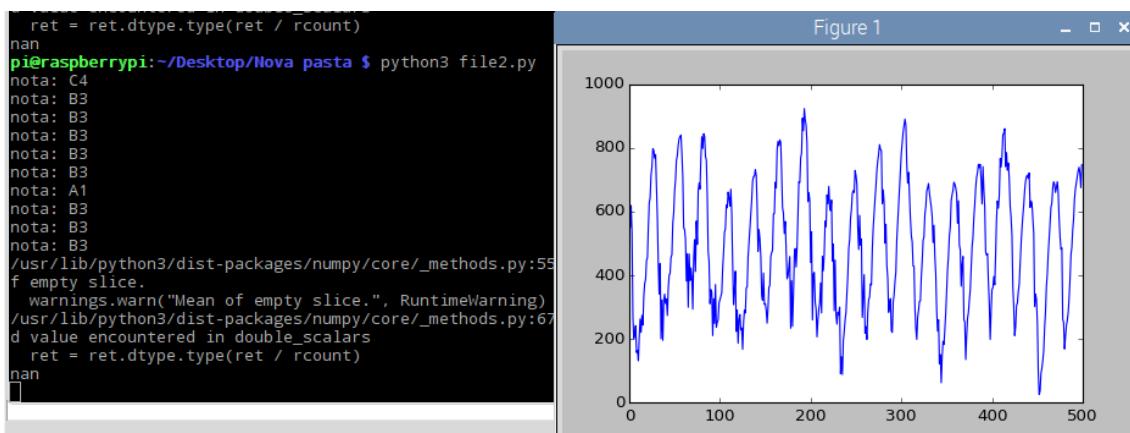


Figura 4.18: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA B3.

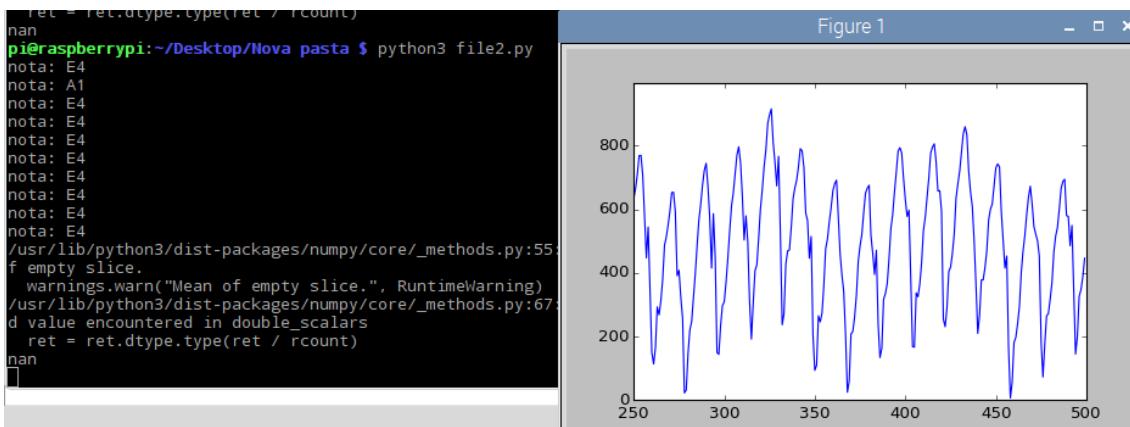


Figura 4.19: RECONSTRUÇÃO E IDENTIFICAÇÃO DA NOTA E4.

Portanto, as imagens anteriores mostram que o experimento apresentou bons resultados, apesar de ser notado pequenas interferências em algumas leituras, que por sua vez quando acontecem, ainda são próximas do resultado esperado para aquele caso.

Logo após terem se encerrado todos os testes, é admissível fazer algumas considerações. Primeiro pode-se denotar que, quanto mais grave um sinal, mais difícil é de efetuar de forma correta a nota correspondente ao mesmo. Isso se deve ao fato de que notas musicais equivalentes às baixas frequências são muito próximas entre si (Tabela 4.1), ou seja, uma pequena variação na frequência do sinal acarreta em uma variação considerável das notas.

Também pode-se ressaltar que o aumento considerável da frequência de amostragem melhora muito a leitura do sinal, no entanto, torna o processo de leitura e identificação um pouco mais lento, sendo assim, é importante trabalhar com esses elementos de forma a otimizar o processo, pois não é desejado que a identificação seja lenta demais, porém é interessante ter uma frequência de amostragem elevada.

5. Conclusões e Sugestões para Trabalhos Futuros

Neste capítulo são apresentadas as conclusões finais do projeto. Além disso, é possível encontrar não só sugestões que podem somar a este trabalho, mas também propostas de desenvolvimento de outros projetos utilizando este como base.

5.1. Conclusões

A primeira parte deste projeto consistiu no projeto e elaboração de um circuito eletrônico analógico com a finalidade de realizar a aquisição de som da melhor forma possível, implementando filtros e realizando ajuste ganho e *off-set* conforme a necessidade. Em seguida foi realizado a elaboração do código, desenvolvido em linguagem Python 3. Este por sua vez, foi usado tanto na configuração dos pinos GPIO, mapeamento das notas, leitura do sensor e com também na aplicação da *FFT* no sinal. Por último, não menos importante, utilizou-se o MCP3008 para comunicação da programação com o circuito eletrônico, de modo que o sinal captador fosse digitalizado dentro dos critérios de reconstrução.

Ao longo do desenvolvimento deste trabalho foram encontradas algumas dificuldades. Uma delas foi lidar com interferências da rede elétrica na leitura do osciloscópio, principalmente em climas chuvosos, que em alguns dias atrapalhou muito a realização de testes intermediários. Como esse ruído possui cerca de 60Hz, é difícil efetuar sua filtragem, mesmo assim, ainda foi possível realizar o projeto, uma vez que em climas mais amenos e interferência ruidosa da rede elétrica nos equipamentos se torna muito pequena, podendo ser relevada.

Outra dificuldade foi selecionar um amplificador operacional adequado para a tarefa realizada neste trabalho. Como o ganho possui um valor elevado, 55V/V, e é dado em um sinal de entrada muito pequeno, isto é, 0.060 V de pico a pico, foi difícil encontrar um modelo capaz para isso, uma vez que amplificadores mais

comuns, por exemplo LM358, não reconheceram a leitura do microfone, isto é, a leitura foi feita como sendo igual a zero.

Sendo assim, através de muitas pesquisas foi identificado o amplificador LM386, o qual tem um circuito interno baseado e ganhos de transistores. Desse modo ele trabalha com ganho de potência, sendo capaz de trabalhar com pequenos sinais e efetuar um ganho entre 26 dB a 46 dB, uma faixa que inclui os valores pretendidos para este trabalho.

O objetivo principal deste trabalho é realizar a implementação de novas tecnologias no âmbito musical e avaliar a eficiência delas. Isso é o que justifica a escolha do Raspberry PI3, um computador de pequeno porte com uma alta capacidade de processamento de dados. Sendo capaz de comunicar esse equipamento com o ambiente externo, abre-se porta para realização de vários estudos e possível criação de novas ideias.

Assim, este trabalho se voltou para o lado musical, por preferência pessoal do autor. O RPI3 ser capaz de realizar leitura de várias frequências analógicas, capturadas por um microfone de eletreto, permite a obtenção de novas ideias, por exemplo, guardar informação de cada sinal identificado em um determinado intervalo de tempo e, posteriormente, resultar em uma sequência de notas. Isso poderia descrever arranjos musicais simples de um instrumento. Deste modo, pode-se dizer que os objetivos propostos foram alcançados.

5.2. Sugestões para Trabalhos Futuros

Como já mencionado, uma vez que o Raspiberry é capaz de reconhecer sinais provenientes de um microfone, muitas outras tarefas podem ser realizadas após essa, além disso, muitas melhorias podem acrescentar neste projeto afim de torna-lo cada vez mais eficiente.

Um ponto que pode acrescentar positivamente nesse trabalho é a elaboração de filtros digitais, visto que, filtros analógicos não podem possuir uma ordem muito elevada, devido a custo e quantidade de componentes. Outro filtro que pode ser implementado digitalmente é um filtro de confiabilidade do resultado,

isto é, após um certo número de testes, ou intervalo de tempo, é resultado a nota que foi identificada uma maior quantidade de vezes.

O filtro de confiabilidade seria um interessante processo a ser acrescentado neste trabalho, pois por meio dele torna-se possível identificar as notas mais frequentes em um intervalo musical simples. Assim, pode-se separar esse intervalo musical em pequenos instantes de tempo e executar o filtro nesses instantes. Desse modo, armazena-se as notas identificadas em cada um destes instantes e, ao término do intervalo musical simples, resulta-se as notas musicais contidas ao longo do intervalo. Com isso, é razoável começar a pensar no conceito de geração de partituras simples em *real-time*. A criação de uma interface para o programa também pode ser elaborada para que essas notas fossem visualizadas mais facilmente.

A Transformada Rápida de *Fourier* (*FFT*) é o método mais conhecido para identificação de frequências e também o mais utilizado. No entanto, existem outros métodos, um deles é a Transformada Q Constante. Ela se consiste em uma otimização da *FFT*, de modo que, requer um menor processamento e faz a identificação da nota musical de forma mais rápida. Essa transformada pode ser estudada e acrescentada neste projeto, com o intuito de substituir a *FFT* implementada. Uma vez que a Transformada Q requer menos processamento do equipamento, pode-se então dedicar parte usar esta diferença de processamento em outras tarefas, ou também aumentar a frequência de amostragem.

Como é visto, são muitas ideias para a continuidade do projeto, comprovando a integridade deste trabalho, o qual abre portas para implementações de novas tecnologias no âmbito musical.

REFERÊNCIAS BIBLIOGRÁFICAS

DOS SANTOS, Cristiano Nogueira. **REPRESENTAC AO ESPECTRAL DE SINAIS PARA TRANSCRICAO MUSICAL AUTOMATICA.** 2004. Tese de Doutorado. UNIVERSIDADE FEDERAL DO RIO DE JANEIRO.

ASTOLA, Jaakko; KUOSMANEN, Pauli. **Fundamentals of nonlinear digital filtering.** CRC press, 1997.

SCHEIRER, Eric D. Tempo and beat analysis of acoustic musical signals. **The Journal of the Acoustical Society of America**, v. 103, n. 1, p. 588-601, 1998.

BACKUS, John; COLTMAN, John W. The acoustical foundations of music. **Physics Today**, v. 23, n. 5, p. 69-70, 1970.

GOHN, Daniel Marcondes. Introdução à Tecnologia Musical. **São Carlos: UFSCAR**, 2012.

MOOG, Robert A. Position and Force Sensors and Their Application to Keyboards and Related Control Devices. In: **Audio Engineering Society Conference: 5th International Conference: Music and Digital Technology.** Audio Engineering Society, 1987.

SCHOLZ, Carter. A proposed extension to the midi specification concerning tuning. **Computer Music Journal**, v. 15, n. 1, p. 49-54, 1991.

UPTON, Eben; HALFACREE, Gareth. **Raspberry Pi user guide.** John Wiley & Sons, 2012.

LUTZ, Mark; ASCHER, David. **Aprendendo Python, 2.** Bookman, 2007.

BORGES, Luiz Eduardo. **Python para Desenvolvedores: Aborda Python 3.3.** Novatec Editora, 2014.

HAYKIN, SIMON S.; VAN VEEN, Barry. **Sinais e sistemas.** Bookman, 2001.

LATHI, Bhagwandas Pannalal. **Sinais e Sistemas Lineares-2.** Bookman, 2006.

ROBERTS, Michael J. **Fundamentos de sinais e sistemas.** AMGH Editora, 2009.

TOCCI, R. J., WIDMER, N. S., MOSS, G. L., **Sistemas Digitais, Aplicações**, 10^a Edição, 2007.

DINIZ, Paulo SR; DA SILVA, Eduardo AB; NETTO, Sergio L. **Processamento Digital de Sinais-: Projeto e Análise de Sistemas**. Bookman Editora, 2014.

NALON, José Alexandre. **Introdução ao processamento digital de sinais**. Grupo Gen-LTC, 2000.

MILETTO, Evandro Manara et al. Introdução à computação musical. In: **IV Congresso Brasileiro de Computação**. 2004.

SEDRA, Adel S., and KENNETH Carless Smith. **Microelectronic circuits**. 5th edition. New York: Oxford University Press, 2006.

BOYLESTAD, RL ATTIA; NASHELSKY, Louis. YAI e KITCHENER, JA **Dispositivos Eletrônicos e Teoria de Circuitos**. 11^a Edição, 2004.

BATES, David J.; MALVINO, Albert Paul. **Eletrônica**. Tradução: José Lucimar do Nascimento. Revisão: Antonio Pertence Júnior. Volume 2. 2007.

PATSKO, Luís Fernando. Aplicações, funcionamento e utilização de sensores. **Maxwell Bohr Instrumentação Eletrônica**, 2006.

LAZZARINI, Victor EP. Elementos de acústica. **Apostila do Departamento de Artes da UEL, Londrina**, 1998.

OGASAWARA, Angélica Soares et al. Reconhecedor de notas musicais em sons polifônicos. **Trabalho de Conclusão de Curso, Graduação em Engenharia Elétrica**. Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2008.

Caderno de Leitura e Escrita Musical I, Escola de Música Villa-Lobos, Rio de Janeiro, Fevereiro, 2007.

CARVALHO, Paulo Cezar et al. **Notas em Matemática Aplicada**. 2009.

LOUREIRO, M. A., PAULA, H. B., **Timbre de um instrumento musical**, Per Musi, Belo Horizonte, n.14, pp. 57-81, 2006.

ADOLFO, Antônio. **O livro do músico**. Irmãos Vitale, 1989.

CHEDIAK, Almir. **Harmonia & Improvisação-Vol. 1**. Irmãos Vitale, 1986.

SCHOENBERG, Arnold. **Fundamentos da composição musical**. Edusp, 1990.

ZUBEN, Paulo. **Música e tecnologia: o som e seus novos instrumentos.** Irmãos Vitale, 2004.

ALVES, Luciano. **FAZENDO MÚSICA NO COMPUTADOR - 2^a EDIÇÃO REVISTA E ATUALIZADA.** Elsevier Brasil, 2006.

LAGO, Nelson Posse. **Processamento Distribuído de Audio em Tempo Real.** 2004. Tese de Doutorado. Universidade de São Paulo.

DE LOURDES SEKEFF, Maria. **Da música-2^a Edição.** Unesp, 2007.

BORCHGREVINK, Hans M. O cérebro por trás do potencial terapêutico da música. **Música e Saúde.** Org. Even Ruud. São Paulo, Summus, p. 57-86, 1991.

MOORE, F. Richard. A technological approach to music. **J. Paynter, T. Howell, R. Or-ton, & P. Seymour, (Eds.) Companion to contemporary musical thought,** v. 1, p. 329-354, 1992.

LOY, Gareth. **Musicians make a standard: The MIDI phenomenon.** Computer Music Journal, US A Vol. I X/4 (winter 1985) p. 8-26. I llustration, bibliography.

FRITSCH, Eloy Fernando. **Música Eletrônica-Uma Introdução Ilustrada.** Eloy Fernando Fritsch, 2008.

Audio Engineering Society In: **Music and digital technology.** New York 1987. p. 109-121.

Notas Musicais. InfoEscola: Navegando e Aprendendo. Disponível em: <<http://www.infoescola.com/musica/notas-musicais/>>. Acesso em: 16 Mai. de 2017.

Scientific Pitch Notation. About.com: Do more. Disponível em: <http://piano.about.com/library/Pitch-Notation/bl_Scientific-Pitch-Notation.htm>. Acesso em: 16 Mai. de 2017.

BREGMAN, Albert S. **Auditory scene analysis: The perceptual organization of sound.** MIT press, 1994.

DA PENHA, Rosani Maria Libardi et al. **ANÁLISE DE SINAIS EM REGIME TRANSIENTE APlicando A TÉCNICA DE WA VELET.** 1999. Tese de Doutorado. UNIVERSIDADE DE SÃO PAULO.

APÊNDICE A

Código para Cálculo das Frequências Existentes em uma Faixa de 7 Oitavas.

```
A4 = 440; %Frquênciade Lá da quarta oitava
st = 2^(1/12); %Tamanho de um Semi-Tom
oitava = st^12;

vt = zeros(12,1);% Tamanho do vetor de um oitava

for i = -9:2
    vt(i+10) = A4*st^i; % Cálculo de frequências da quarta oitava
end

mt = zeros(12,7); % Quantidade de oitavas

for i = -3:3
    mt(:,i+4) = vt*oitava^i; % Cálculo de todas as frequências em 7 oitavas
end
```

Figura 1: CÓDIGO PARA CÁLCULO DE FRQUÊNCIAS.

APÊNDICE B

Fotos da montagem real do circuito eletrônico.

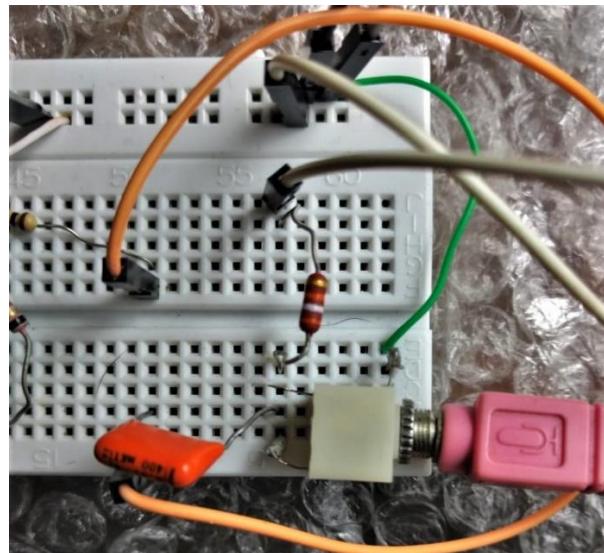


Figura 2: MONTAGEM DO MICROFONE.

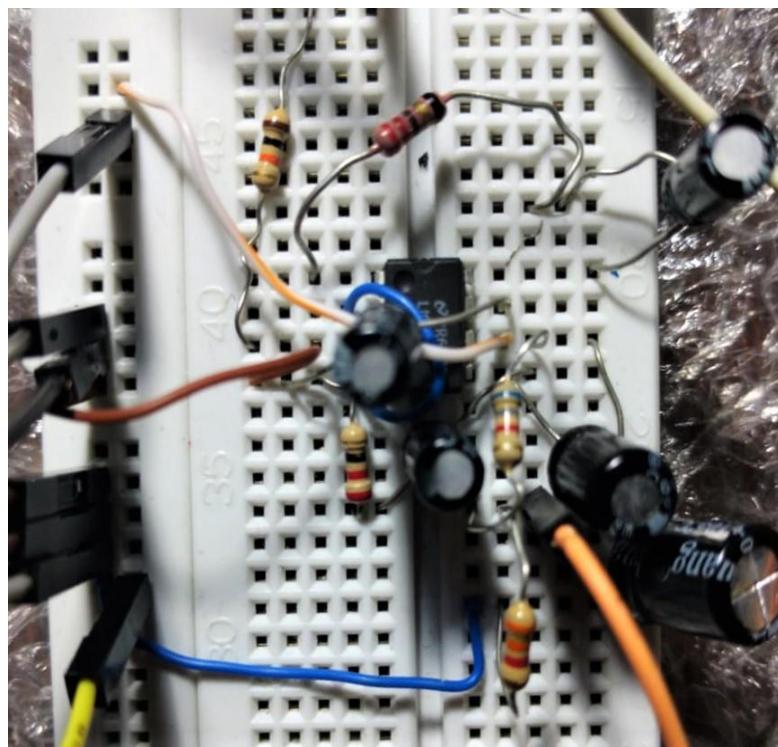


Figura 3: MONTAGEM CIRCUITO DE GANHO E OFF-SET.

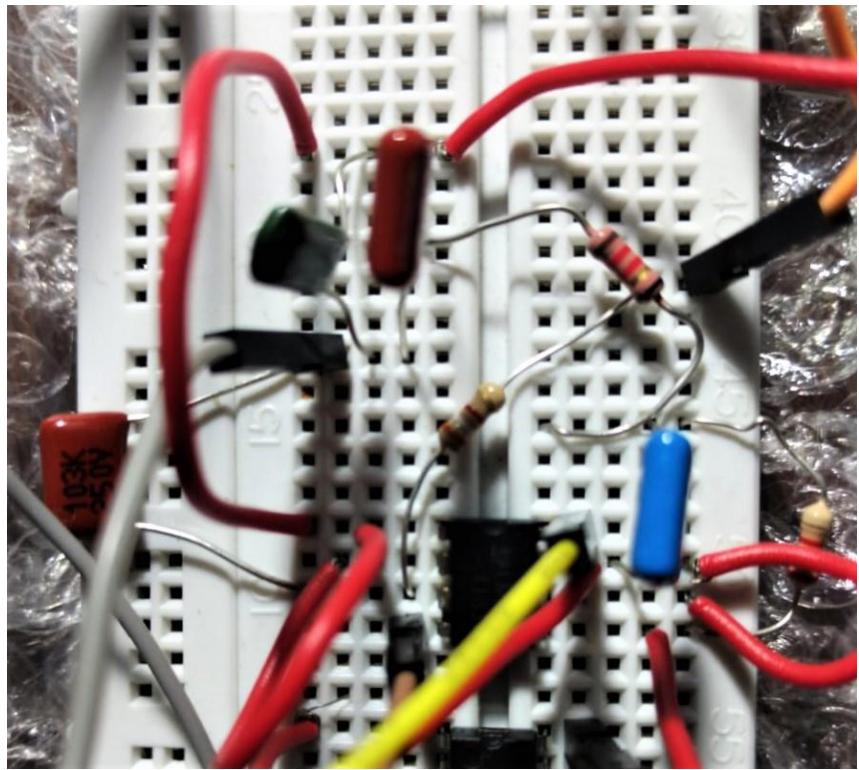


Figura 4: MONTAGEM DO FILTRO ATIVO DE TERCEIRA ORDEM.

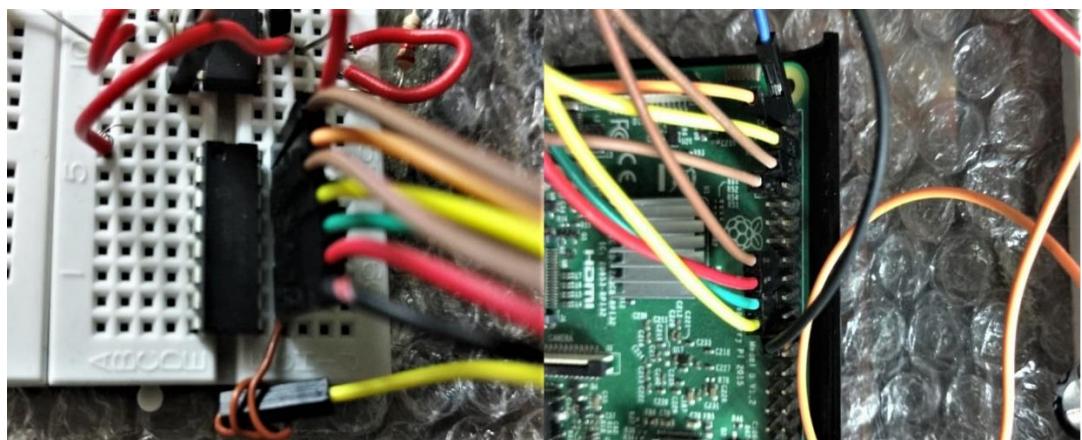


Figura 5: LIGAÇÃO ENTRE O CONVERSOR MCP3008 E AS GPIO SPI.

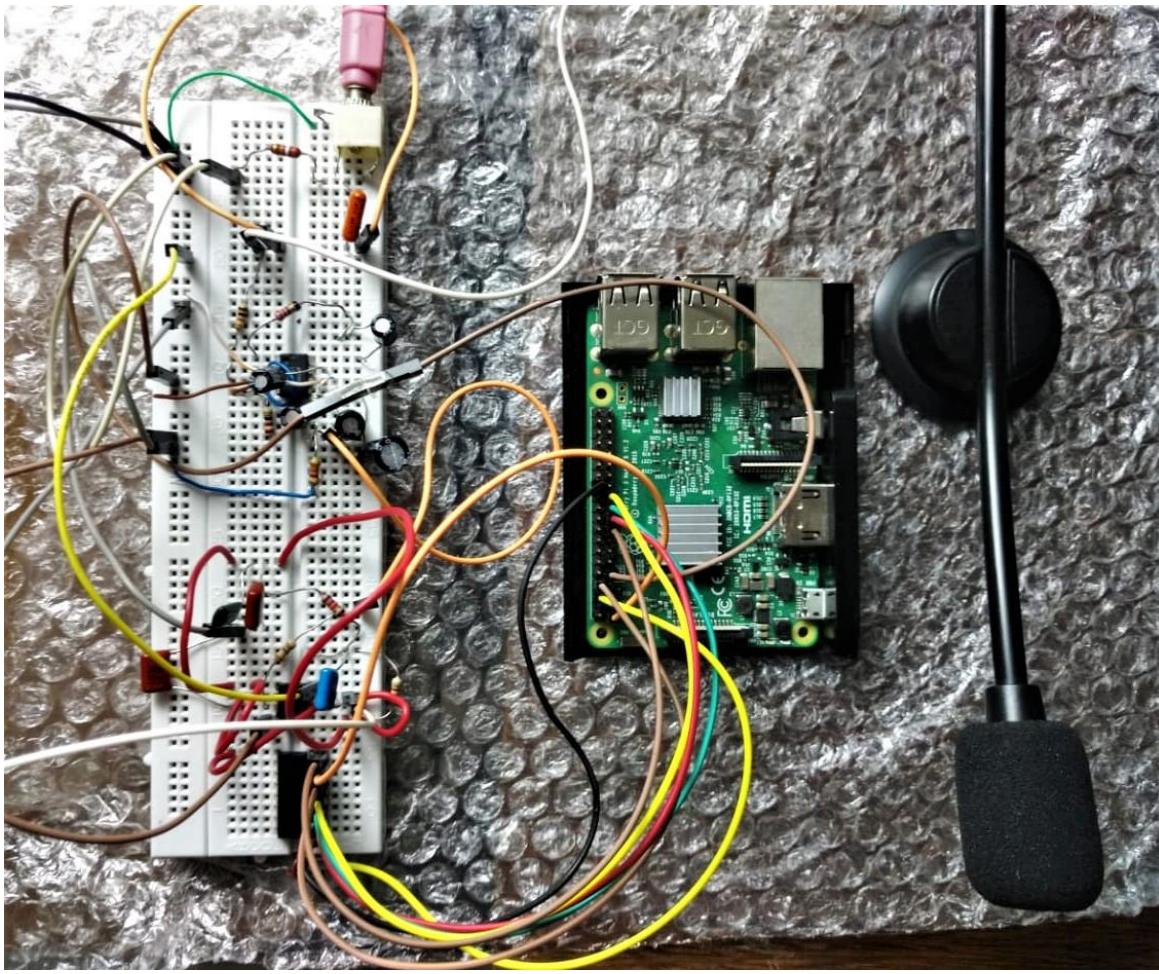


Figura 6: MONTAGEM GERAL DO CIRUITO ELETRÔNICO COMPLETO.

APÊNDICE C

Código utilizado para implementação de *FFT*, relação de notas musicais e geração de gráficos.

```
file2.py *  
1 import numpy as np #biblioteca para utilizar FFT  
2 import scipy as sp #biblioteca para utilizar FFT  
3 import scipy.fftpack #biblioteca para utilizar FFT  
4 import time  
5 import matplotlib.pyplot as plt #biblioteca para fazer plotagens  
6  
7 import Adafruit_GPIO.SPI as SPI #biblioteca para utilizar portas SPI  
8 import Adafruit_MCP3008 #biblioteca para utilizar MCP3008  
9  
10 SPI_PORT = 0  
11 SPI_DEVICE = 1  
12 mcp = Adafruit_MCP3008.MCP3008(spi=SPI.SpiDev(SPI_PORT, SPI_DEVICE)) #Configuracao dos pinos SPI para leitura do MCP3008  
13  
14 # Aplicação da FFT no sinal  
15 def spectrum(y,Fs):  
16     n = len(y) # comprimento do sinal  
17     k = np.arange(n) # separa valores uniformemente espaçados  
18     T = n/Fs # calcula o periodo de amostras  
19     frq = k/T # numero de amostras  
20     frq = frq[np.arange(int(n//2))] #one side frequency range  
21     Y = sp.fftpack.fft(y)/n # calculo e normalizacao da FFT  
22     Y = Y[np.arange(int(n//2))] #analisa por comprimento a FFT  
23     return frq, abs(Y) # Retorna o valor absoluto do sinal  
24  
25 Fs = 15000 #Frequencia de amostragem
```

Figura 7: Inclusão Das Bibliotecas ---- Configuração Dos Pinos SPI ---- Função Responsável Pelo Cálculo Da *FFT*.

```
24  
25 Fs = 15000 #Frequencia de amostragem  
26 Ps = 100 #Numero de pontos adquiridos  
27 ys = [] # cria lista vazia  
28 ts = [] #  
29 fs = [] #  
30  
31 for _ in range(Ps): #Armazenamento de um ciclo do sinal no vetor  
32     y = mcp.read_adc(0) #realiza leitura do conversor  
33     ys.append(y) # armazena os valores no em ys  
34  
35 k = 0  
36 while True: #Condicao de parada caso seja desejado  
37     k+=1  
38     if k > 10000:  
39         break  
40  
41     ts.append(time.time()) # armazena tempo de execucao em ts  
42     y = mcp.read_adc(0) #le conversor  
43     ys.append(y) # armazena valores em ys  
44     ys = ys[-Ps:] # define ys com um n'umero limitado de pontos  
45  
46 #Identificação da nota
```

Figura 8: Definição Da Frequência De Amostragem E Número De Pontos Amostrados ---- Armazenamento Dos Valores Oditidos Na Leitura Do Conversor ---- Estabelece Condição De Parada ---- Releitura Do Conversor E Atualização Dos Valores.

```

45
46     # Identificação da nota
47     if k % Ps == 0:
48         Fs = 1/np.mean(np.array(ts[1:]) - np.array(ts[:-1])) #cria um ciclo de amostra
49         ts = [] #atualiza ts para lista vazia
50         f,a = spectrum(ys,Fs) # computa a FFT do sinal
51         f=f[1:] #retorna frequencia
52         a=a[1:] #retorna amplitude
53         i = np.where(a == max(a))
54         mf = np.asscalar(f[i]) # define valor absoluto como frequencia identificada
55         fs.append(mf) #acrescenta esse valor a lista fs
56         fs = fs[-10:]
57         if k % (10*Ps) == 0:
58             mapa = np.fromfunction(lambda i, j: (2)**(j-3) * 440*(2**((1/12))**((i-9), (12, 7))) #cria mapa de notas com
59                                         #base na matematica musical
60             notas = ['C','C#/Db','D','D#/Eb','E','F','F#/Gb','G','G#/Ab','A','A#/Bb','B'] # nomeia as notas
61
62             if not mapa[0,0] < mf < mapa[-1,-1]: # limita mapa ate a ultima nota
63                 #print('NOTA NAO MENSURADA')
64                 continue
65
66             i,j = np.array(np.where(mapa.flat[np.abs(mapa - mf).argmin()] == mapa)).flatten() #relaciona valor
67                                         #de frequencia com a nota e sua altura
68
69             nota = notas[i] #retorna a nota
70             altura = j+1 # retorna a altura da nota
71
72             print('nota: %s%s'%(nota,altura)) #mostra a nota na tela do terminal
73
74     print(np.mean(np.array(ts[1:]) - np.array(ts[:-1])))
75
76     plt.plot(range(len(ys)), ys)
77     plt.show() #plotagem dos ultimos pontos do sinal

```

Figura 9: ENVIO DO SINAL AMOSTRADO PARA FUNÇÃO RESPONSÁVEL PELA FFT ---- RETORNA O VALOR ABSOLUTO DE FREQUÊNCIA OBTIDO NA FFT ---- COMPARA VALOR ABSOLUTO COM OS VALORES DAS FREQUÊNCIAS MAPEADAS ---- RETORNA A NOTA E ALTURA CORRESPONDENTE ---- PLOTA O GRÁFICO DO SINAL AMOSTRADO.